

Goddard 2001 IR&D Engineering Document

Ocean Color Model Inversion Using an Analog, Artificial Neural Network

Steven A. Bailey
GSFC
Code 584
March 29, 2002

Table of Contents

Introduction	3
Objectives.....	3
Neural Network Software Architecture.....	4
Hardware Design.....	6
Active AANN	6
Passive AANN	10
Summary	12
Conclusions	13
Appendices	15
Appendix A – Matlab Source Code	15
Appendix B – Microchip Source Code	39
Appendix C - Tables	54
Appendix D - Coefficients	68
Appendix E - Photos	69
Appendix F - Schematics	73
Appendix G - Printed Circuit Boards (PCBs).....	83

Introduction

This research was conducted under the auspices of Flight and Science Information Systems for the Goddard 2001 IR&D program. The intention was to design and build an Analog Artificial Neural Network (AANN) that will invert or calculate a single ocean color constituent from ocean color spectra detector voltages as input.

Spectral data recorded from the Airborne Diode Array Spectrometer (ADAS) was used to train an AANN. This inversion technique was further refined by reducing ADAS data from 256 channels to five channels. These five channels match those found in the first five of eight channels of the Sea-viewing Wide Field-of-view Sensor (SeaWiFS). Channel centers are at 412, 443, 490, 510, and 555 nanometers (see Figure 2).

A simple design was created using only operational amplifiers, resistors, and diodes. It was determined through simulation (using PSPICE) and prototyping that voltage dividers could accurately represent neural network weights. Also discovered was small signal diodes arranged in a clamping configuration can accurately represent a sigmoid squashing function. The squashing function is the basis of all multilevel perceptron (MLP) neural networks.

A successful attempt to refine this circuit lead to a passive design requiring no active (opamps) components. This circuit is composed solely of resistors and diodes. Since no amplification is possible, input voltages must be large because this circuit is essentially an attenuator. As seen in Figures 13 and 14, output closely matches that of the digital neural network output of Figure 6.

Objectives

The specific objectives of this research are multifaceted. The foremost objective was to develop a neural network software architecture which could accurately invert ocean color spectra into a single constituent...namely chlorophyll concentration. To do this required collecting and normalizing ocean color spectra recorded by ADAS. A flight on October 6, 1997 was chosen for several reasons. First, this flight contained several patches of high chlorophyll concentration which was determined by the Airborne Oceanographic Lidar (AOL). Second, this data consists of three separate flight lines which collectively were used for neural network training (flight line 'D' was used for training while flight lines 'B' & 'C' were used for validation). Finally, this data provided ground truth for a SeaWiFS overpass shown in Figure 1.

The next objective was to take this software architecture and reduce it to a minimum configuration. This was desired to reduce the amount of hardware needed to implement this design.

The third objective was to simulate this minimum configuration in a circuit simulator called PSPICE. This was an important and useful step to verify whether the design would (or should) work as planned.

The fourth and final objective was to build and verify the designed circuit via prototype printed circuit boards (PCBs). This step also required additional custom hardware used for verification (which was built under this proposal).

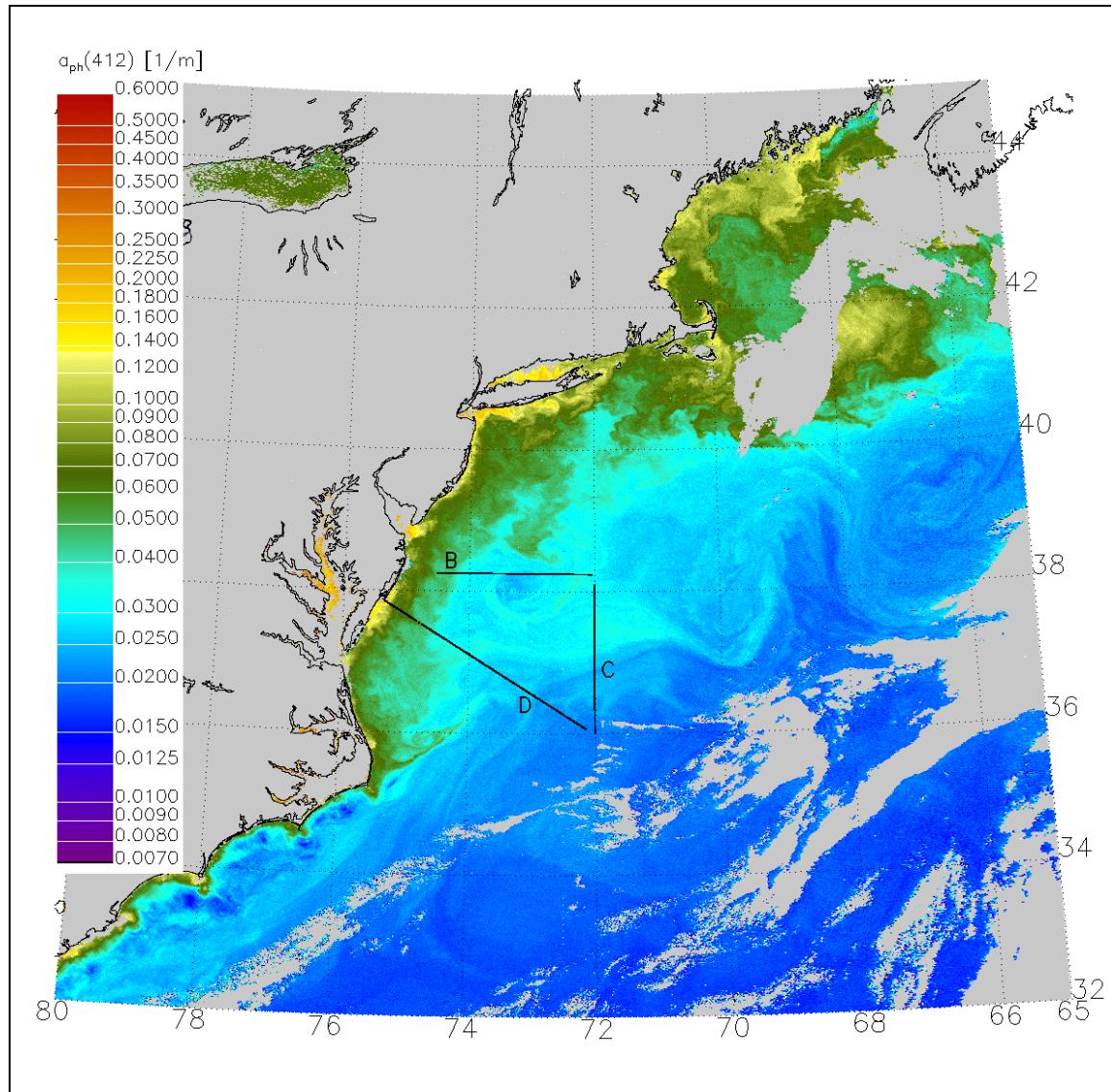


Figure 1 - SeaWiFS image from Oct. 6, 1997 superimposed with ADAS flight lines

Neural Network Software Architecture

Once an adequate network was trained using all 256 channels of ADAS data, I reduced the network ultimately to five channels. This is the minimum number of channels that could be used and still obtain usable inversion results. The first five of eight SeaWiFS

channels were chosen centered at 412, 443, 490, 510, and 555 nanometers. Since this AANN was going to be implemented in hardware, fewer channels equated to less hardware (Figure 2). In AANN nomenclature, this design is a 5x2x1 Multilayer Perceptron (MLP) with five inputs, two hidden nodes, and one output.

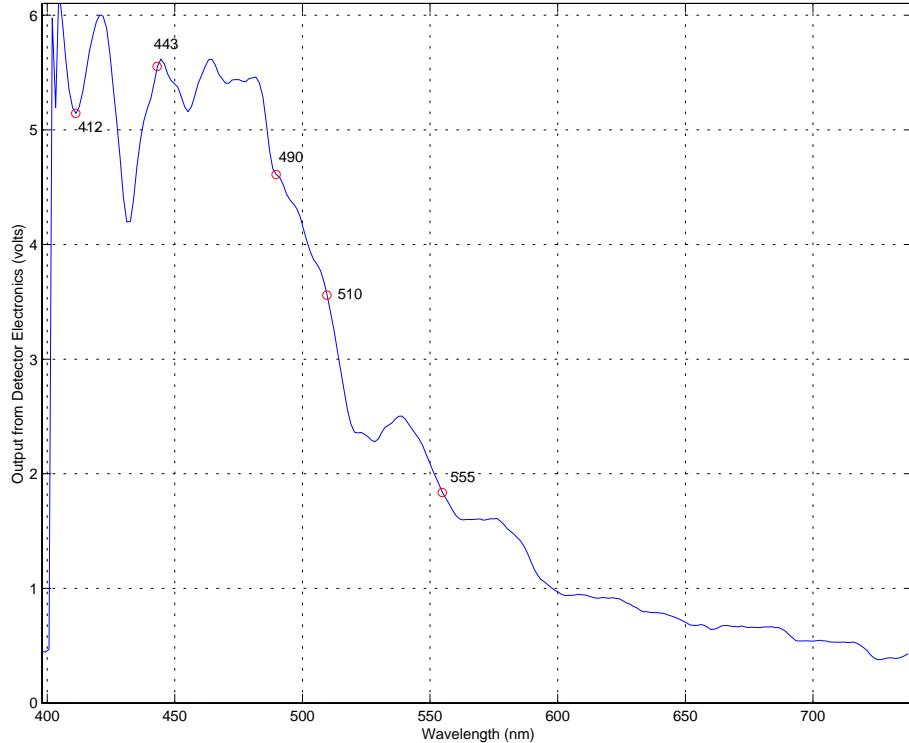


Figure 2 - Typical ADAS spectra showing 5 SeaWiFS channels used to train AANN

Using Matlab (version 5.1) with the Neural Network Toolbox (version 2.0.4), a 5x2x1 neural network was trained. The hidden nodes used the tansig squashing function which is sigmoidal and symmetrical (about zero). The single output used no squashing function. It simply was the inner product of the two hidden nodes and subsequent weights and biases (Figure 3). It should be noted that hardware was made to approximate the tansig squashing function. In the end, the hardware was directly modeled and incorporated into Matlab as custom squashing functions. This led to a higher correlation between Matlab (digital) results and those found using the AANN.

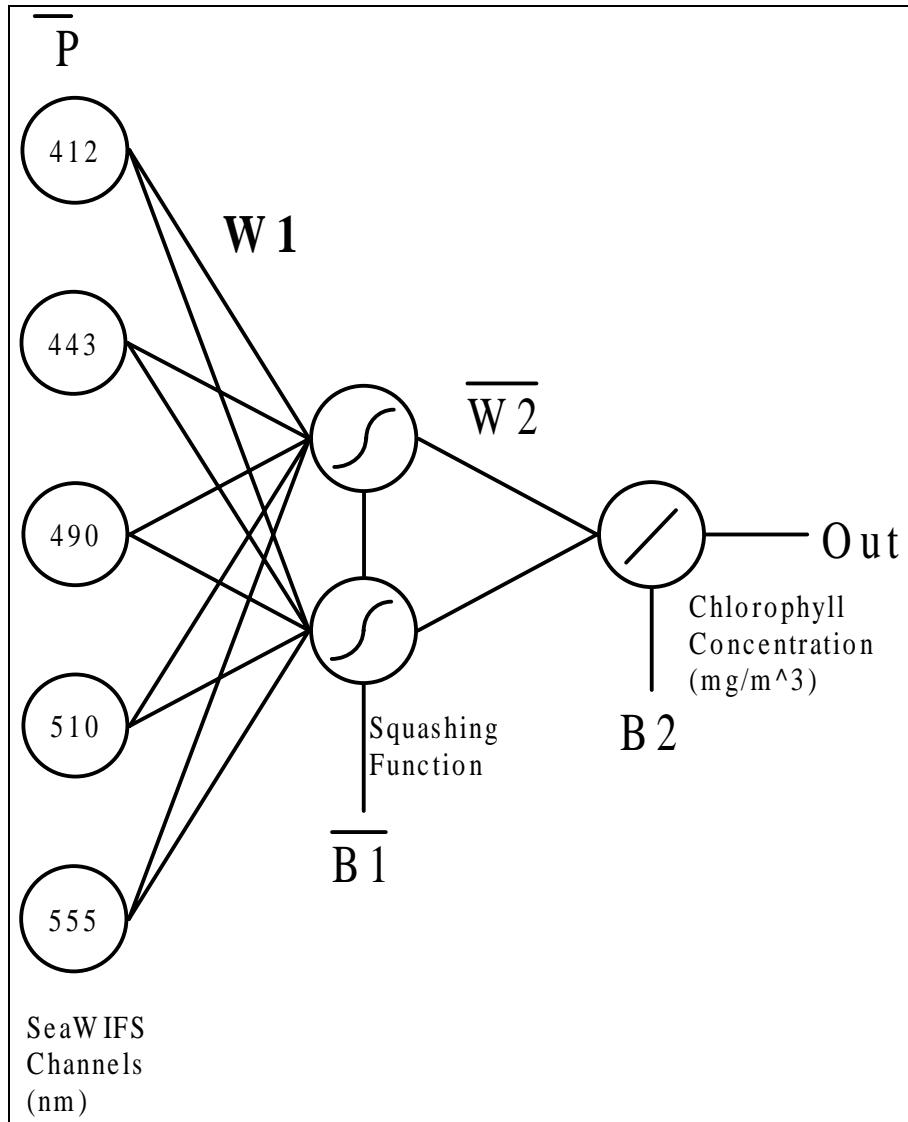


Figure 3 - AANN node diagram

Hardware Design

Active AANN

At this stage, work began using a circuit simulator called PSPICE. This software allowed creation (in simulation) of neural network circuits composed of electronic components. The first circuit is composed of three components: Operational amplifiers, resistors, and diodes. The operational amplifiers are used as buffers (voltage followers), inverters, and as adders. Resistors are used in a voltage divider arrangement to represent weights of the network. Keeping weights within the range of -1 to 1 , these voltage dividers act as multipliers. Finally, small signal diodes arranged as voltage clamps provide the squashing function of the network. When inputs to the voltage clamps are low, the diodes do not conduct and the circuit acts linearly. As voltage rises, conduction begins in an

exponential manner eventually becoming asymptotic. This transfer function turns out to be very similar to the Matlab ‘tansig’ transfer function as seen in Figure 5.

This circuit is an active AANN (or AAANN for short). A subset of the design is found in Figure 4. Within this figure, the neural network matrix operations are displayed under different stages of the circuit. The final equation for this circuit and subsequent circuits is found in Figure 10. The complete AAANN design can be found in Figures 21-27.

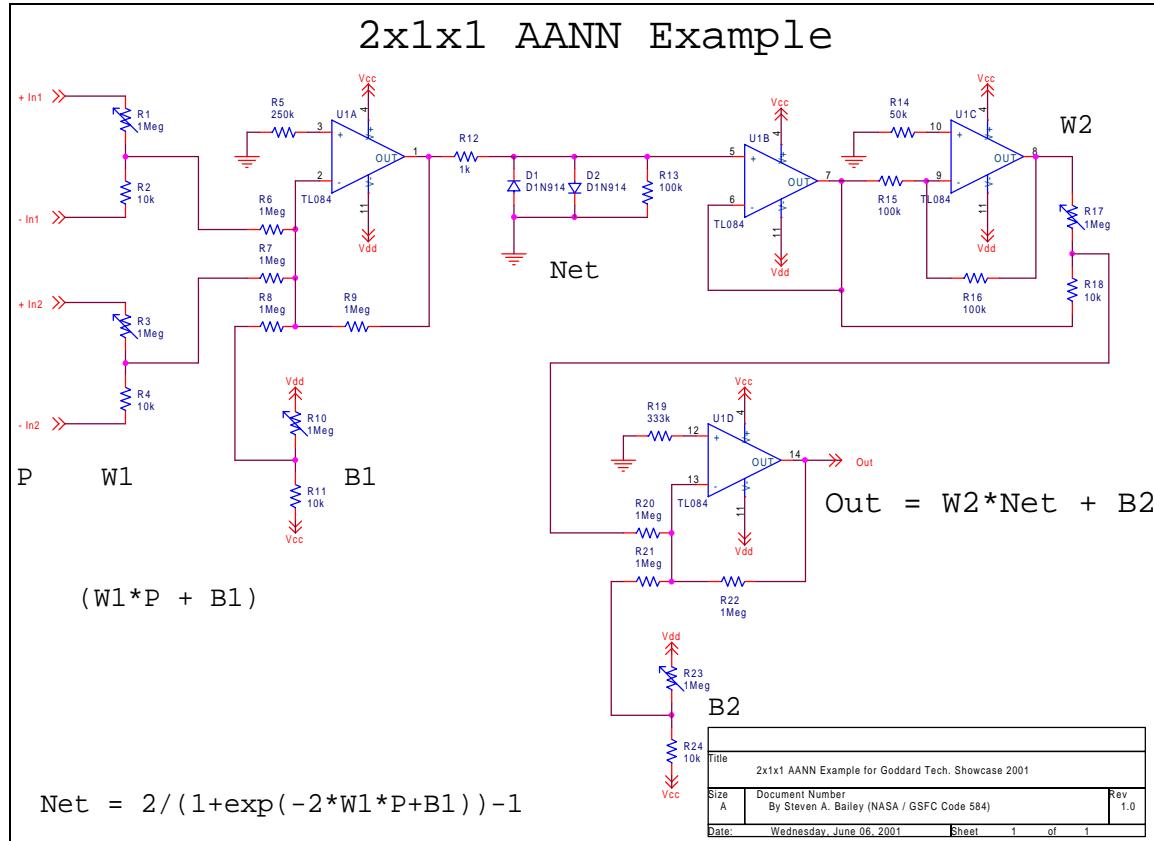


Figure 4 - Subset of final AANN design

After simulating this circuit, a prototype board was created and populated using inexpensive opamps, potentiometers, and small signal diodes (Figure 28). Basic functionality of the circuit was verified using a function generator driving each input one at a time. Since the network was originally represented digitally using Matlab, a simulation of function generator input was made both in Matlab and Pspice. Results were then compared to actual output of the circuit via an oscilloscope. There was strong agreement.

To better represent the ‘true’ response of the diodes in their clamping role, an empirical model of their response was developed. Using a voltage source, voltages were manually stepped from -9 to 9 volts, recording circuit response. Below (Figure 5) is found a superposition of three plots comprised of the Matlab, ‘tansig’ squashing function, the active AANN squashing function, and finally a passive AANN squashing function

(described in the next section). All three models coincide linearly up until around ± 0.5 volts input. From there, all three models differ. Since each model must be differentiable everywhere (by convention when training back propagation neural networks), the second plot shows differences in their slopes. Noise found in the 'Green' and 'Blue' plots are artifacts due to the transitions from the three polynomials that were pieced together to represent these functions. When used in neural training, these artifacts have insignificant effects.

At this point, a modification to the training algorithm was needed to accommodate input weights restricted to a range of -1 to 1 and the use of our empirical transfer function. This work can be found in the file 'trainneural_5ch_measure.m' under the 'Source Code', 'Matlab' section.

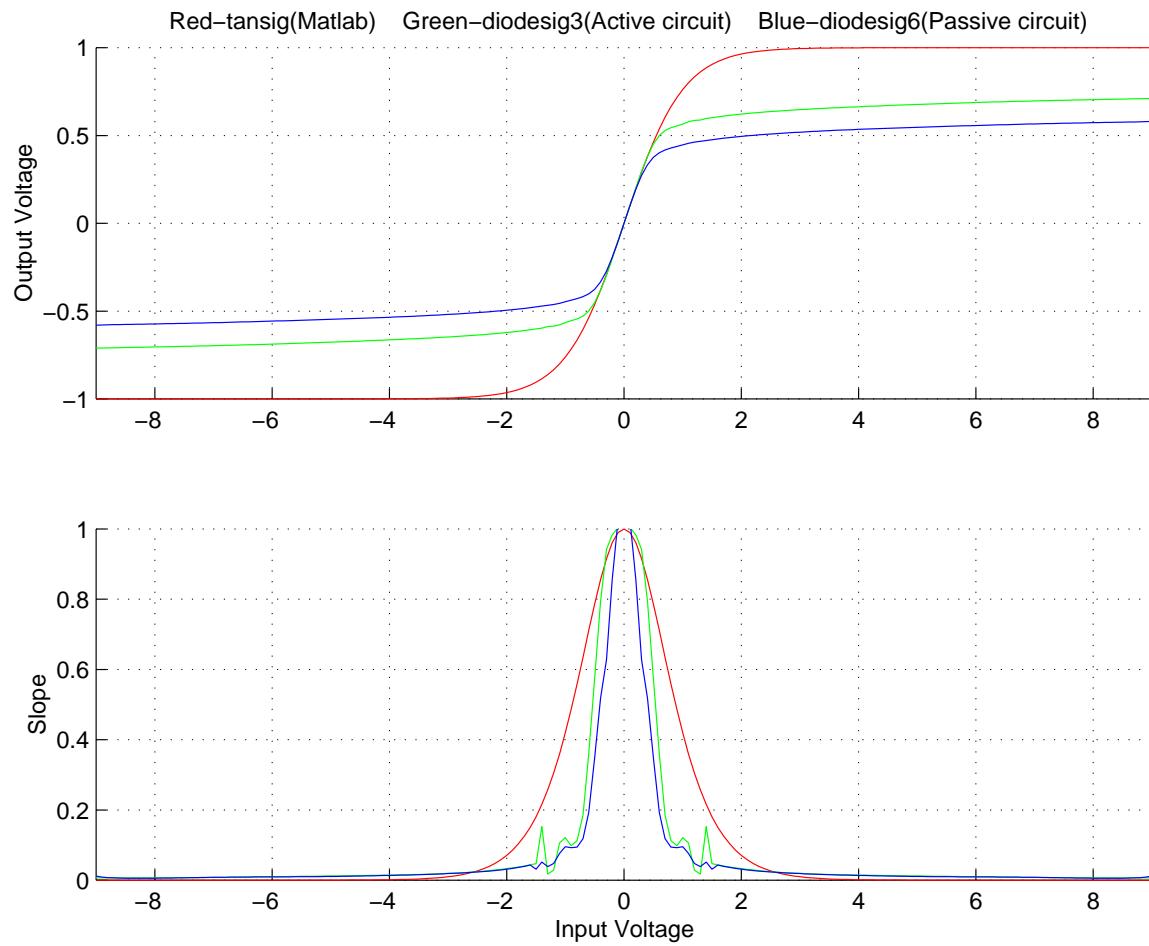


Figure 5 - Transfer functions and slopes

To test five channels of spectral data with this circuit, a digital-to-analog (DAC) circuit was designed to act as a front-end driver to the AAANN (Figures 18-20, 30). 100 5-tuple data points were recorded in DAC memory from Oct. 7, 1997, flight line 'D'. This data was normalized prior to storage in the DAC board. The DAC outputs spectral data in a 10 millisecond loop. Each 5-tuple vector remained stable on the AAANN input for 100

microseconds. This loop time was a limitation of the DAC board. As seen in Figure 10, the AAANN circuit has two orders of magnitude more bandwidth than the DAC imposed loop time.

Test results using the DAC board show strong agreement between AAANN output and Matlab output (Figure 6). Figures 11 and 12 are oscilloscope photographs showing AAANN output using flight line 'D' as input. Figure 6 is a Matlab plot using the same flight line as input. It shows the five input channels vs. sample number in the top box. The bottom box shows the actual chlorophyll concentration (rescaled) in blue superimposed with the digital neural network output in red. Again, there is strong agreement between the analog and digital implementations of this neural network.

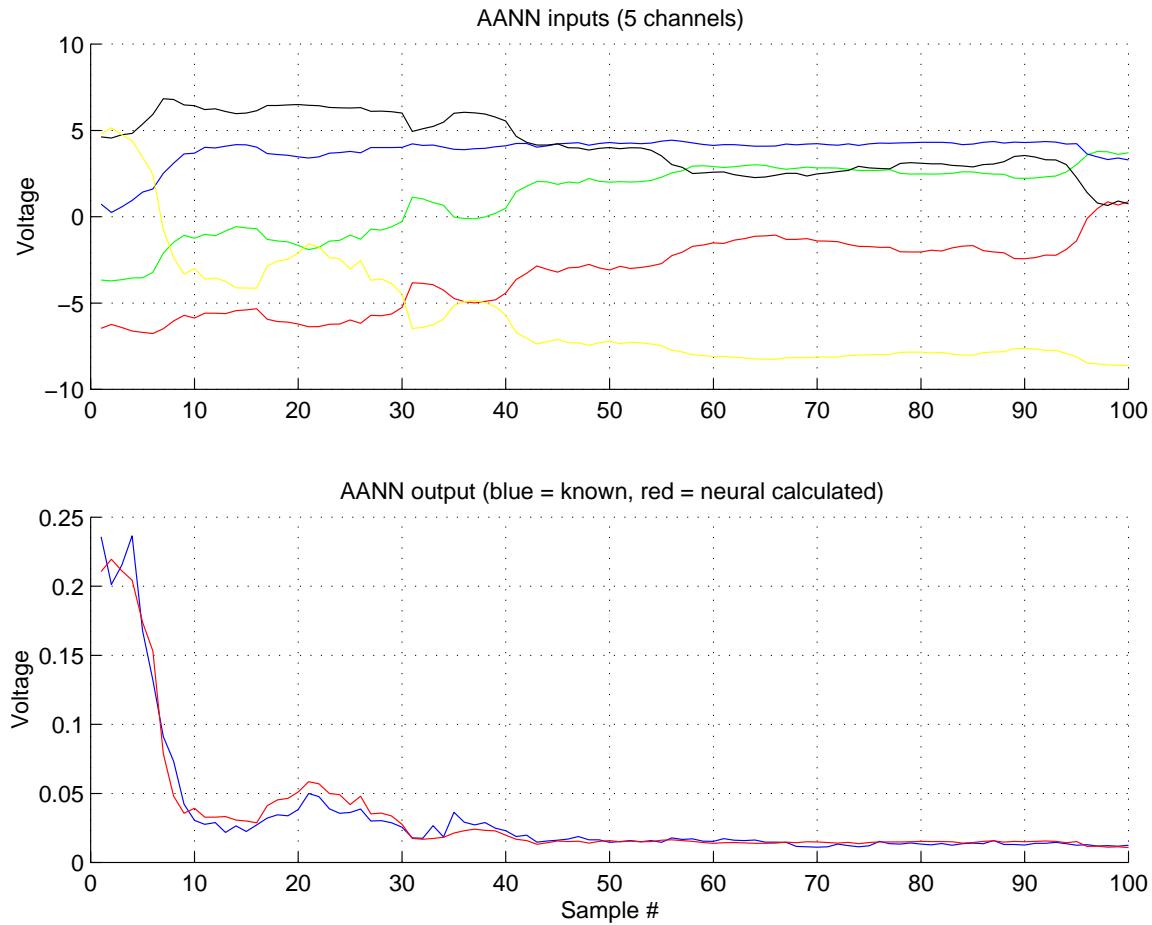


Figure 6 - AANN inputs (top) AANN output (bottom)

Figure 7 is a Matlab plot showing the output (known chlorophyll concentration) of the three flight lines superimposed with output from the software representation of the active AANN. The network was trained on flight line 'D' (top plot) and validated on flight lines 'B' and 'C' (bottom two plots).

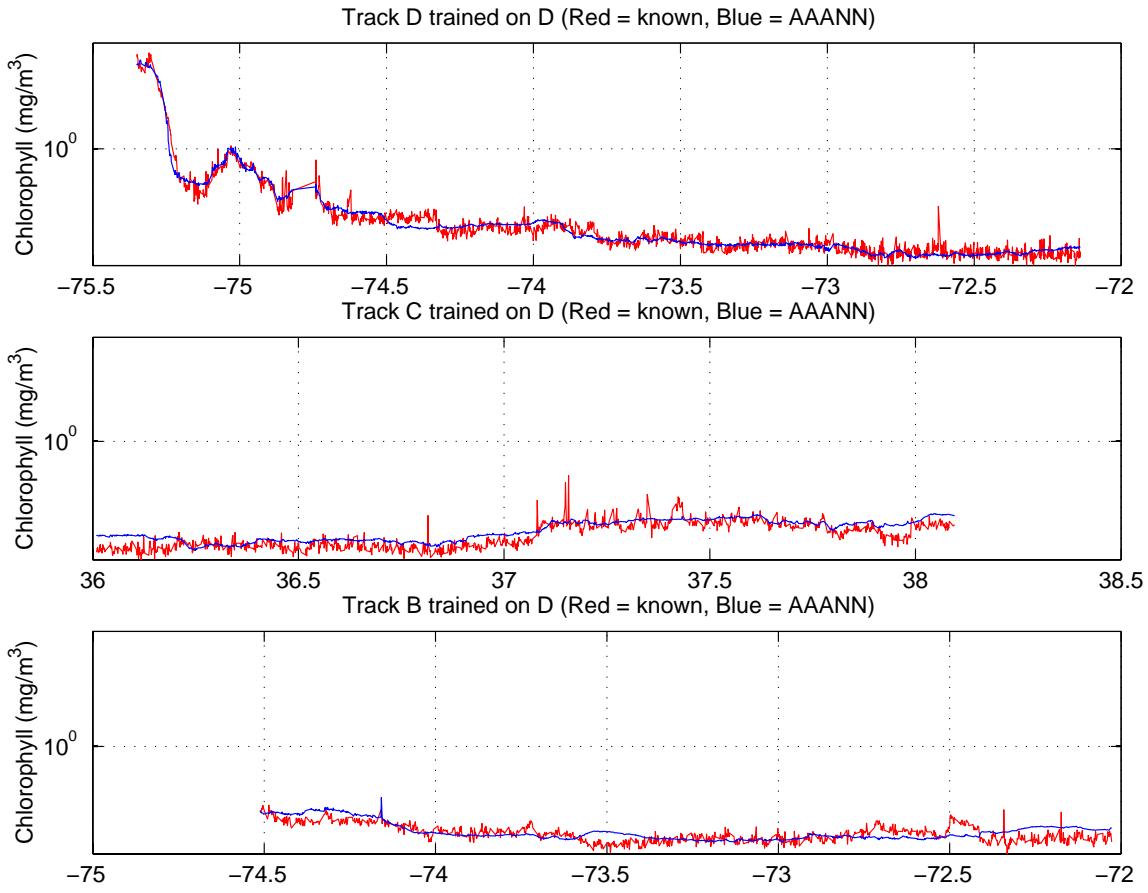


Figure 7 - Oct. 7, 1997 flight lines superimposed with AAANN results in red. Network was trained on flight line ‘D’ and validated with flight lines ‘B’ and ‘C’.

Passive AANN

Having success with the AAANN, an all-passive version was designed called the passive AANN (or PAANN for short). This device would be composed of only resistors and diodes with no opamps present. Obviously, no buffering exists in this circuit. Therefore, passive buffering was implemented by increasing the impedance from one stage of the circuit to another. Although not as precise as the active circuit, this method worked given the small size of our network.

Weights in the first layer would be represented by voltage dividers not unlike the AAANN. However, to reduce attenuation, both second layer weights are forced to one (Figure 8). Therefore, this configuration requires large input voltages on the order of -9 to 9 volts/channel.

Like before, a prototype board was created after simulation in PSPICE (Figure 29). As evident in Figure 8, this is a simple circuit. It requires few parts of which none are active. To realize the same inversion as did in the AAANN, a more constrained and elaborate training algorithm had to be developed. Instead of training our network by adjusting all weights and biases, we needed an algorithm that would converge to a solution where

output weights remain static at 1. Also, due to our single input voltage dividers acting as input weights, weights here had to remain between 0 and 1. Finally, another empirical transfer function was generated by measuring the response of our clamping diodes in the circuit. Therefore, both a custom training algorithm and new transfer function were developed. The ensemble of this work can be found in the file ‘trainneural_5ch_914_cir3_posw.m’ under ‘Appendix A - Source Code’.

A custom designed DAC was again used to drive this circuit. The only difference being input voltages had to be scaled higher than the AAANN. Figures 13 and 14 are oscilloscope photographs showing PAANN output using flight line ‘D’ as input. These photographs show strong agreement with the AAANN derived Matlab plot (bottom) of Figure 6.

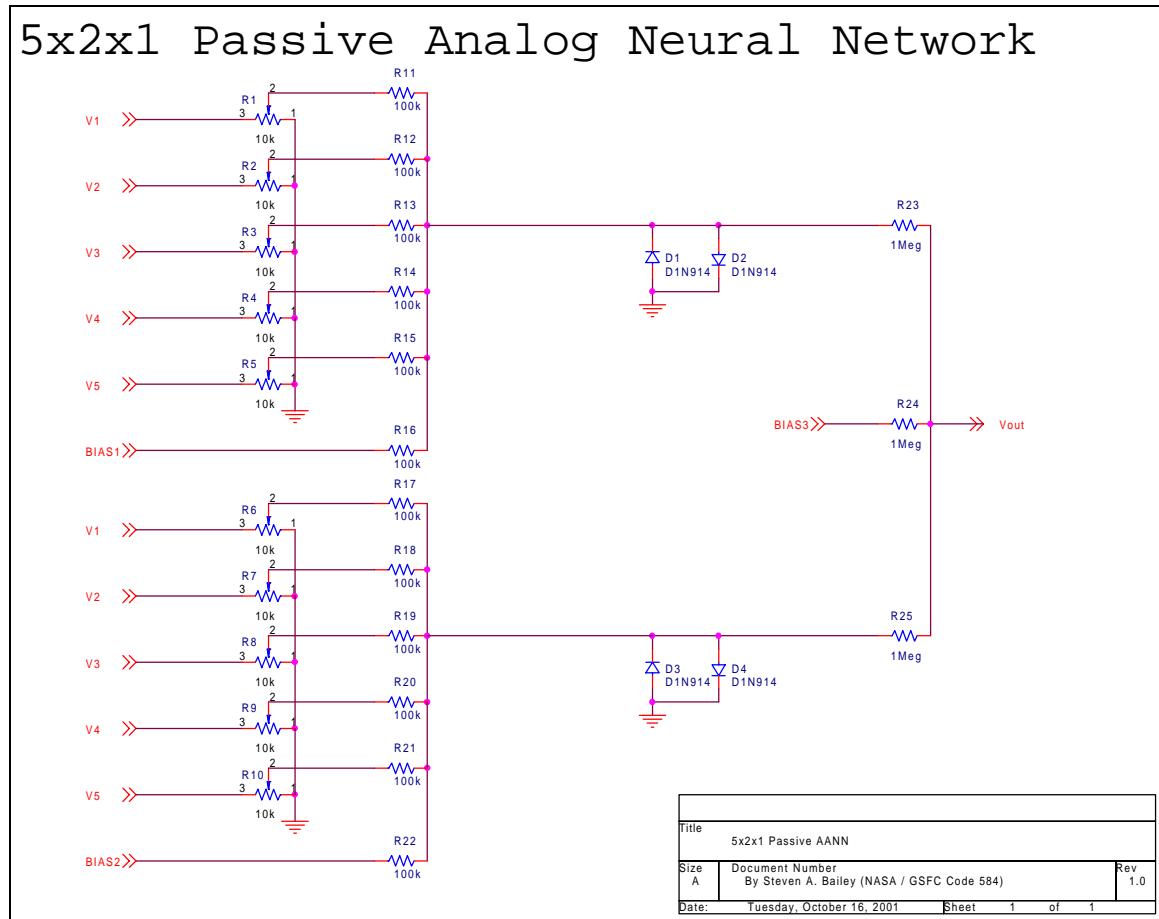


Figure 8 - Passive AANN Schematic

Figure 9 is a Matlab plot showing the output (known chlorophyll concentration) of the three flight lines superimposed with output from the software representation of the passive AANN. The network was trained on flight line ‘D’ (top plot) and validated on flight lines ‘B’ and ‘C’ (bottom two plots).

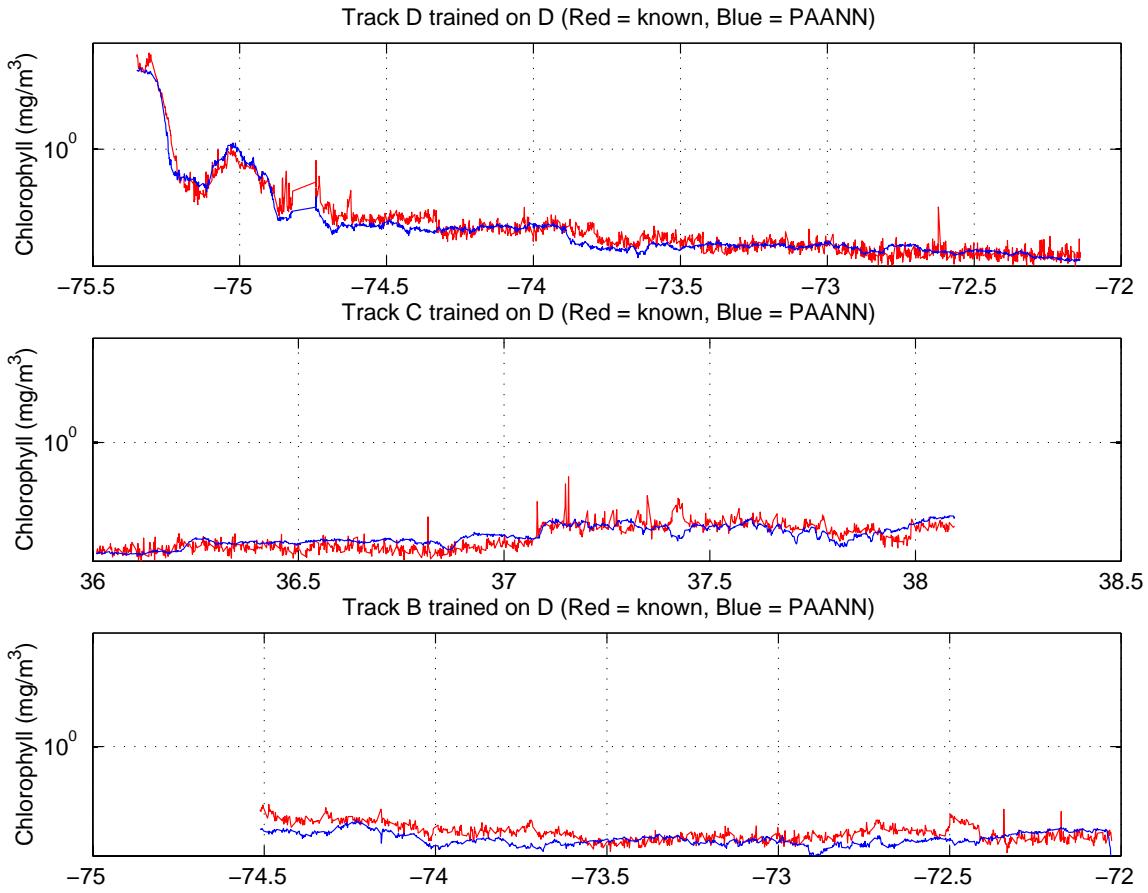


Figure 9 - Oct. 7, 1997 flight lines superimposed with PAANN results in red. Network was trained on flight line ‘D’ and validated with flight lines ‘B’ and ‘C’.

Summary

Figure 10 shows the digital equation of our neural network and gives approximate calculation times for a single instance of this equation across different technologies. The Pentium III processor, running at 500 Mhz, is well suited for many tasks including those which are math intensive like this. The Microchip 16C65A is a digital microcontroller typically used in embedded applications. It runs at 20 Mhz and is orders of magnitude slower than the Pentium. The Microchip also draws much less power than the Pentium, so we are comparing quite different hardware devices. The Active AANN has a bandwidth based on its opamps characteristics and how they are configured. With this 5x2x1 neural arrangement, calculation times are only slightly slower than the Pentium. The Passive AANN has a bandwidth based on the capacitive coupling of passive components. Calculation times are still quite fast with a power consumption well below that of the Active AANN. Calculation times and power consumption could have been reduced for both circuits had more specialized components been used.

$$\text{Out} = \mathbf{W2} * \left[\frac{2}{1 + e^{(-2 * \mathbf{W1} * \mathbf{P} + \mathbf{B1})}} - 1 \right] + \mathbf{B2}$$

Processor	Type	Bandwidth	Power	Calc. Time
Pentium III	Digital	500 Mhz	35 Watts	616 nsec.
PIC16C65A	Digital	20 Mhz	1.0 Watt	5 msec.
Active AANN	Analog	4 Mhz	1.0 Watt	1 usec.
Passive AANN	Analog	NA	0.1 Watt	10 usec.

Figure 10 - Artificial neural network equation with corresponding processing times

Conclusions

There are always pros and cons to every design decision. This research is no exception. Artificial neural networks are certainly not the only answer to difficult or intractable problems such as ocean color model inversion. However, they are a viable solution given their potential compact design, robustness, and noise immunity.

Disadvantages of using AANNs are just as numerous as advantages. Analog circuits are subject to bias shifts. These shifts are mainly due to temperature changes which vary the internal properties of both semiconductors and resistors. Resistors arranged as voltage dividers (as a whole) are not affected, but semiconductors are. These bias shifts cause network calculations to vary. Another disadvantage is network weights must be changed mechanically by turning a potentiometer. Programmable digital potentiometers do exist, but time and resources did not allow experimentation with them. Another disadvantage is that some type of sample-and-hold circuit must be built to feed the AANN if the signal in question is acquired serially (like the ADAS is). Although not complicated, this additional circuitry does increase the complexity of the AANN. Another problem for this example was spectral voltages had to be normalized before the AANN. This normalization was performed before digital data was stored in the DAC. For a real-time application, normalization would have to be performed by analog means. This would not be trivial. Finally, building a 5x2x1 AANN can only be used with problems that work in this domain. Although this circuit could easily be scaled up or down, it would require a redesign of the hardware.

Results clearly show that neural networks (in general) can successfully invert ocean color data. These results also indicate that simple analog hardware can implement said network if signals (voltages) are available simultaneously as inputs to the AANN. Through the use of both simulation and prototyping, it was found that an AANN is quite fast when compared with conventional digital processors. Based on the network equation of Figure 10, the active AANN is a slightly slower than a Pentium III microprocessor running at 500 Mhz. Of note is the electrical power required by each device. A 500 Mhz Pentium III consumes on average 35 Watts of power. The un-optimized active AANN consumes 1

Watt while the passive AANN consumes 0.1 Watts. The PIC16C65A is a digital microcontroller that was included for comparison. It too draws little electrical power, but is orders of magnitude slower than either AANN due to its digital composition.

As a final note, an AANN does have utility in sensors where multichannel input data is available simultaneously. This simultaneous input would remove the need for sample-and-hold circuitry. In addition, problems not requiring input normalization would also simplify the circuitry allowing for direct connection between said sensor channel output and AANN input. In this ideal configuration, analog calculations could be available microseconds after sensor acquisition thereby requiring no digital intervention.

Appendices

Appendix A – Matlab Source Code

The following Matlab code is used to make Figure 2.

```
%-----  
% showspectra.m  
%  
% This a Matlab script designed to display a typical ADAS spectrum along with  
% 5 SeaWiFS channels superimposed. Data is first rescaled back into engineering  
% units (volts) to be consistant with this topic of AANNs.  
%  
% S.A.B.      NASA      10/26/2001  
%-----  
  
load 003853ad.cal_new;                      % Load ADAS calibration file  
load 181222fl_adas_aol.bin.mat;            % Load data file  
s      = size(Radiance);                     % Get size of Radiance file  
  
Radiance = Radiance ./ (X003853ad(:,3)*ones(1,s(2)));    % Rescale Radiance back  
plot(X003853ad(:,2),(Radiance(:,1)./4096).*10);        % into voltage  
  
hold on; grid;  
plot(X003853ad([11 35 70 85 119],2), (Radiance([11 35 70 85 119],1)./4096).*10, 'ro');  
axis([398 740 0 (2500/4096)*10]);                % Rescale axis  
  
 xlabel('Wavelength (nm)');                    % Add text  
 ylabel('Output from Detector Electronics (volts)');  
 title('ADAS Spectrum');  
 text(0.4121e3,(2.0597e3/4096)*10,'412');  
 text(0.4418e3,(2.3540e3/4096)*10,'443');  
 text(0.4907e3,(1.9526e3/4096)*10,'490');  
 text(0.5138e3,(1.4629e3/4096)*10,'510');  
 text(0.5580e3,(0.8067e3/4096)*10,'555');
```

The following Matlab code is used to make Figure 6.

```
%-----  
% digital3_100.m  
%  
% This Matlab script serves two purposes. It first creates a file of input voltages  
% which will be uploaded to EEPROM of the 2nd DAC board. This board 'plays' these  
% voltages in an endless loop to either the active AANN or the passive AANN. The  
% second purpose is to create two Matlab subplots of input voltages and resulting  
% output volages.  
%  
% S.A.B.      NASA      10/26/2001  
%-----  
  
% Load neural coefficients  
load 181222fl_coeffsl_div20_5ch_914_posw3.mat; % Where (0<=W1<=1) & (W2=1)  
  
load 003853ad.cal_new; % Load ADAS calibration file  
  
load 181222fl_adas_aol.bin.mat; % Load flight spectra  
Chloro = Chloro.*3; % Convert to mg/m^3  
s = size(Chloro); % Get size of Chlorophyll  
Radiance = Radiance ./ (X003853ad(:,3)*ones(1,s(2))); % Rescale to voltage  
  
P = Radiance([11 35 70 85 119],:); % Reduce to 5 SeaWiFS channels  
  
% The following line does the  
% following...it normalizes P  
% (Radiance) by first subtracting  
% the mean (producing data with  
% zero mean) and then dividing by  
% the standard deviation (producing  
% data with a variance of one).  
% A final multiplication (by 5)  
% produces voltages large enough  
% to drive passive AANN.  
  
I = ones(5,1); M = I*mean(P); SD = I*std(P); P = (P-M)./SD; P = P.*5;  
  
Dset = P(:,2309:-11:1210); % Create 100 points for EEPROM  
Known = Chloro(:,2309:-11:1210)./20; % Create corresponding point of Chloro  
% Calculate neural output based on  
% passive AANN. Take note we are using  
% 'diodesig6' as hidden node transfer  
% function. Passive circuit does a  
% division by 6 in the input stage  
% and a division by 3 in the output  
% stage...hence the divisors.  
Neural = simuff(Dset./6,W1,B1./6,'diodesig6',W2,B2,'purelin')./3;  
  
Output = Dset.*204.75 + 2047.5; % Put in digital form where  
% Analog range is -10 to 10  
% where -10 corresponds to 0 &  
% 10 corresponds to 4095  
  
Output = round(Output); % Round to nearest integer  
  
fid = fopen('digital3_100.txt','w'); % Open file for writing  
  
for i=1:100 % Must commutate for EEPROM.  
    fprintf(fid, '\tda\t%d\t%f\t%f\n', Output(1,i),Known(i),Neural(i));  
    fprintf(fid, '\tda\t%d\t%f\t%f\n', Output(2,i),Known(i),Neural(i));  
    fprintf(fid, '\tda\t%d\t%f\t%f\n', Output(3,i),Known(i),Neural(i));  
    fprintf(fid, '\tda\t%d\t%f\t%f\n', Output(4,i),Known(i),Neural(i));  
    fprintf(fid, '\tda\t%d\t%f\t%f\n', Output(5,i),Known(i),Neural(i));  
end  
  
fclose(fid); % Close file  
subplot(2,1,1); % First subplot displays 5 input
```

```

grid on;                                % channels
hold on;
plot(Dset(1,:),'r');
plot(Dset(2,:),'g');
plot(Dset(3,:),'b');
plot(Dset(4,:),'k');
plot(Dset(5,:),'y');
title('AANN inputs (5 channels)');
ylabel('Voltage');

subplot(2,1,2);                         % Second subplot displays known
grid on;                                % Chlorophyll output superimposed
hold on;                                 % with neural calculated output.
plot(Known);                            % Known output
plot(Neural,'r');                       % Neural output
title('AANN output (blue = known, red = neural calculated)');
xlabel('Sample #');
ylabel('Voltage');

```

The following Matlab code is used to make Figure 5.

```
%-----  
% calc3funcs.m  
%  
% This program displays three neural network transfer functions used in this  
% study which are:  
%  
% 1. tansig - Matlab transfer function commonly used in the digital world.  
% 2. diodesig3 - Transfer function measured in active AANN circuit.  
% 3. diodesig6 - Transfer function Measured in passive AANN circuit.  
%  
% It also displays three derivatives of said functions which are:  
%  
% 1. deltatan - Derivative of 'tansig'.  
% 2. deltadiode3 - Derivative of 'diodesig3'.  
% 3. deltadiode6 - Derivative of 'diodesig6'.  
%-----  
% S.A.B.           NASA           10/26/2001  
%-----  
  
x = -9:.1:9;                                % Set input range  
subplot(2,1,1);                               % Create top plot  
hold on; grid on;  
plot(x,tansig(x),'r');                      % Plot Matlab 'tansig' function  
plot(x,diodesig3(x),'g');                   % Plot measured active function  
plot(x,diodesig6(x),'b');                   % Plot measured passive function  
axis([-9 9 -1 1]);                          % Force axis  
title('Red-tansig(Matlab)      Green-diodesig3(Active circuit)      Blue-diodesig6(Passive  
circuit)');  
ylabel('Output Voltage');  
  
subplot(2,1,2);                               % Create bottom plot  
hold on; grid on;  
plot(x,deltatan(tansig(x)),'r');            % Plot Matlab 'tansig' derivative  
plot(x,deltadiode3(diodesig3(x)),'g');     % Plot measured active derivative  
plot(x,deltadiode6(diodesig6(x)),'b');     % Plot measured passive derivative  
axis([-9 9 0 1]);                           % Force axis  
  
xlabel('Input Voltage');                     % Plot text  
ylabel('Slope');
```

The following Matlab code is used to make Figure 7.

```
%-----
% calcreal_AAANN_5ch.m
%
% This program displays the three flight lines from Oct. 7, 1997 superimposed
% with active AAANN (AAANN) results.
%-----
% S.A.B.          NASA          10/26/2001
%-----

load 181222fl_coeffs_div10_2_5ch.mat;           % Neural coeffs.
                                                % trained on 181222fl_adas_aol.bin.mat
load 003853ad.cal_new;                         % Load ADAS calibration file

load 181222fl_adas_aol.bin.mat;                % Load flight line 'D'
Chloro      = Chloro.*3;                        % Convert to mg/m^3
s          = size(Chloro);                      % Get size of data
                                                % Recover raw counts
Radiance = Radiance ./ (X003853ad(:,3)*ones(1,s(2)));
P          = Radiance([11 35 70 85 119],:);       % Reduce to 5 channels

                                                % Normalize data
I = ones(5,1); M = I*mean(P); SD = I*std(P); P = (P-M)./SD;

                                                % Compute neural calc.
Neural = simuff(P,W1,B1,'diodesig3',W2,B2,'purelin').*10;

figure;                                         % Create figure
subplot(3,1,1);                                % Create top plot

semilogy(Lon,Chloro,'r');                      % Creat120
e known plot in red
grid on; hold on;
semilogy(Lon,Neural,'b');                       % Create neural plot in blue
a = axis;
axis([a(1) a(2) 0 6]);                         % Force axis

ylabel('Chlorophyll (mg/m^3)');                 % Add text
title('Track D trained on D (Red = known, Blue = AAANN)');

%-----

load 173608fl_adas_aol.bin.mat;                % Load flight line 'C'
Chloro      = Chloro.*3;                        % Convert to mg/m^3
s          = size(Chloro);                      % Get size of data
Radiance = Radiance ./ (X003853ad(:,3)*ones(1,s(2))); % Recover raw counts
P          = Radiance([11 35 70 85 119],:);       % Reduce to 5 channels

                                                % Normalize data
I = ones(5,1); M = I*mean(P); SD = I*std(P); P = (P-M)./SD;

                                                % Compute neural calc.
Neural = simuff(P,W1,B1,'diodesig3',W2,B2,'purelin').*10;

subplot(3,1,2);                                % Create middle plot

semilogy(Lat,Chloro,'r');                      % Create known plot in red
grid on; hold on;
semilogy(Lat,Neural,'b');                       % Create neural plot in blue
a = axis;
axis([a(1) a(2) 0 6]);                         % Force axis

ylabel('Chlorophyll (mg/m^3)');                 % Add text
title('Track C trained on D (Red = known, Blue = AAANN)');

%-----

load 165123fl_adas_aol.bin_b.mat;              % Load flight 'B'
Chloro      = Chloro.*3;                        % Convert to mg/m^3
```

```

s      = size(Chloro);                      % Get size of data
Radiance = Radiance ./ (X003853ad(:,3)*ones(1,s(2))); % Recover raw counts
P      = Radiance([11 35 70 85 119],:);       % Reduce to 5 channels

% Normalize data
I = ones(5,1); M = I*mean(P); SD = I*std(P); P = (P-M)./SD;

% Compute neural calc.
Neural = simuff(P,W1,B1,'diodesig3',W2,B2,'purelin').*10;

subplot(3,1,3);                           % Create bottom plot

semilogy(Lon,Chloro,'r');                % Create known plot in red
grid on; hold on;
semilogy(Lon,Neural,'b');                 % Create neural plot in blue
a = axis;
axis([a(1) a(2) 0 6]);                  % Force axis

ylabel('Chlorophyll (mg/m^3)');           % Add text
title('Track B trained on D (Red = known, Blue = AAANN)');

```

The following Matlab code is used to make Figure 9.

```
%-----
% calcreal_PAANN_5ch.m
%
% This program displays the three flight lines from Oct. 7, 1997 superimposed
% with passive AANN (PAANN) results.
%-----
% S.A.B.          NASA          10/26/2001
%-----

% Neural coeffs.
% trained on 181222fl_adas_aol.bin.mat

load 181222fl_coeffsl_div20_5ch_914_posw3.mat;

load 003853ad.cal_new; % Load ADAS calibration file

load 181222fl_adas_aol.bin.mat; % Load flight line 'D'
Chloro = Chloro.*3; % Convert to mg/m^3
s = size(Chloro); % Get size of data
% Recover raw counts
Radiance = Radiance ./ (X003853ad(:,3)*ones(1,s(2)));
P = Radiance([11 35 70 85 119],:); % Reduce to 5 channels

% Normalize data
I = ones(5,1); M = I*mean(P); SD = I*std(P); P = (P-M)./SD; P = P.*5;

% Compute neural calc.
Neural = simuff(P./6,W1,B1./6,'diodesig6',W2,B2,'purelin')./3;

figure; % Create figure
subplot(3,1,1); % Create top plot

semilogy(Lon,Chloro,'r'); % Create known plot in red
grid on; hold on;
semilogy(Lon,Neural.*18); % Create neural plot in blue
a = axis;
axis([a(1) a(2) 0 6]); % Force axis

ylabel('Chlorophyll (mg/m^3)'); % Add text
title('Track D trained on D (Red = known, Blue = PAANN)');
%-----

load 173608fl_adas_aol.bin.mat; % Load flight line 'C'
Chloro = Chloro.*3; % Convert to mg/m^3
s = size(Chloro); % Get size of data
% Recover raw counts
Radiance = Radiance ./ (X003853ad(:,3)*ones(1,s(2)));
P = Radiance([11 35 70 85 119],:); % Reduce to 5 channels

% Normalize data
I = ones(5,1); M = I*mean(P); SD = I*std(P); P = (P-M)./SD; P = P.*5;

% Compute neural calc.
Neural = simuff(P./6,W1,B1./6,'diodesig6',W2,B2,'purelin')./3;

subplot(3,1,2); % Create middle plot

semilogy(Lat,Chloro,'r'); % Create known plot in red
grid on; hold on;
semilogy(Lat,Neural.*18); % Create neural plot in blue
a = axis;
axis([a(1) a(2) 0 6]); % Force axis

ylabel('Chlorophyll (mg/m^3)'); % Add text
title('Track C trained on D (Red = known, Blue = PAANN)');
%-----
```

```

load 165123fl_adas_aol.bin_b.mat; % Load flight line 'B'
Chloro = Chloro.*3; % Convert to mg/m^3
s = size(Chloro); % Get size of data
% Recover raw counts
Radiance = Radiance ./ (X003853ad(:,3)*ones(1,s(2)));
P = Radiance([11 35 70 85 119],:); % Reduce to 5 channels

% Normalize data
I = ones(5,1); M = I*mean(P); SD = I*std(P); P = (P-M)./SD; P = P.*5;

% Compute neural calc.
Neural = simuff(P./6,W1,B1./6,'diodesig6',W2,B2,'purelin')./3;

subplot(3,1,3); % Create bottom plot

semilogy(Lon,Chloro,'r'); % Create known plot in red
grid on; hold on;
semilogy(Lon,Neural.*18); % Create neural plot in blue
a = axis;
axis([a(1) a(2) 0 6]); % Force axis

ylabel('Chlorophyll (mg/m^3)'); % Add text
title('Track B trained on D (Red = known, Blue = PAANN)');

```

The following Matlab code is used to train a neural network that will work with our active circuit. The active circuit was our first design of three.

```
%-----
% trainneural_5ch_measure.m
%
% This program trains on the Inbound Track from the Oct. 6, 1997 flight.
% This Track was chosen because of its high variability. The following neural
% network file of trained coefficients was produced:
%
% 181222f1_coefficients_div10_2_5ch.mat
%
% This version uses 'diodesig3.m' and 'deltadiode3.m' which were
% derived from measured data of a ln914 diode. This measured data can be found in
% file 'pin914_measure.mat'.
%
% This code was developed to train a neural network that could be implemented in
% an active circuit (circuit1). This circuit uses only opamps and voltage dividers
% to represent both weights and biases. In this circuit, our weights (W1 and W2) are
% kept between -1 and 1. This allows us to use fewer opamps since no amplification is
% needed within this domain.
%
% We use the standard Levenberg-Marquardt for training. The idea with this code
% is we present as input, spectra that is scaled large enough to realize a network
% with weights between -1 and 1. In other words, our solution space is broadened with
% this empirically determined scaling factor.
%-----
% S.A.B.           NASA          05/30/2001
%-----

load 181222f1_adas_aol.bin.mat;           % Oct. 6, 1997 inbound track
load 003853ad.cal_new;                   % Load ADAS calibration file
Chloro    = Chloro.*3;                   % Convert to mg/m^3

s        = size(Chloro);                 % Get size of file
Radiance = Radiance ./ (X003853ad(:,3)*ones(1,s(2))); % Recover raw counts
P        = Radiance([11 35 70 85 119],:); % Reduce to 5 channels centered at
                                         % 412, 443, 490, 510, and 555 nm

M        = mean(P);                     % Used to normalize data with zero mean
SD      = std(P);                      % and unity std.

I        = ones(5,1);                   % Vectorize for direct calculation
M        = I*M;
SD      = I*SD;

P        = (P - M)./SD;                % Normalize spectra as described above
%-----

half   = floor(s(2)/2);                % Calc. half of data
ch     = Chloro./10;                  % Divide chlorophyll by 10
index = randperm(s(2));                % Create random index the size of data
rch    = ch(index);                  % Create shuffled data sets
rP     = P(:,index);

%
% This part trains on 3000 pts. from the 'shuffled' deck and
% then does a verification on another (different) 3000 pts. from
% same deck. When verification error stops decreases or Threshold is met,
% training stops.
%-----

Threshold = .001;                      % Stop training at this threshold
while(1)
    % Initialize weights and biases
```

```

[W1,B1,W2,B2] = initff(P,2,'diodesig3',ch,'purelin');
OldError = 1234567e45;

while(1)
    % Make a training run of 50 iterations
    [W1,B1,W2,B2,TE,TR] = trainlm(W1,B1,'diodesig3',W2,B2,'purelin',
        rP(:,1:half),rch(1:half),[10 50 Threshold]);

    % Test training coefficients with other
    % half of data
    Test = simuff(rP(:,half+1:s(2)),W1,B1,'diodesig3',W2,B2,'purelin');
    Diff = Test-rch(half+1:s(2));

    Error = sumsqr(Diff);           % Calc. sum squared error on test set
    % Print current error
    fprintf('Test Error = %f\n\n', Error);

    if (Error >= OldError)          % If test error is increasing, exit loop
        W1 = OldW1; B1 = OldB1;
        W2 = OldW2; B2 = OldB2;
        break;
    end

    if (TR <= Threshold)           % If training error reaches threshold,
        break;                      % then quit
    end

    OldError = Error;
    OldW1 = W1; OldB1 = B1;
    OldW2 = W2; OldB2 = B2;
end                                     % If, after exiting from above...both W1
                                         % and W2 (weights) are in the domain of
                                         % -1 to 1, then exit outside loop and
                                         % we are done training. By searching for
                                         % solutions where weights are constrained
                                         % to this domain, we can represent these
                                         % weights in hardware with simple voltage
                                         % dividers.

if ( (max(max(abs(W1))) <= 1.0) & (max(max(abs(W2))) <= 1.0) )
    break;
end

```

end

The following Matlab code is used to train a neural network that will work with our passive circuit. The passive circuit was our third design of three.

```
%-----
% trainneural_5ch_914_cir3_posw.m
%
% This program trains on the Inbound Track from the Oct. 6, 1997 flight.
% This Track was chosen because of its high variability. The following neural
% network file of trained coefficients was produced:
%
% 181222f1_coeffsl_div20_5ch_914_posw3.mat
%
% This version uses 'diodesig6.m' and 'deltadiode6.m' which were
% derived from measured data of a ln914 diode while in circuit3. This measured
% data can be found in file 'dln914_circuit3.mat'.
%
% This code was developed to train a neural network that could be implemented in
% a passive circuit (circuit3). This circuit uses no opamps. Only voltage dividers
% are used to represent both weights and biases. In this particular circuit, our
% output layer weights (W2) are fixed at 1.0 requiring no dividers. Input layer
% weights (W1) are held positive to a domain between 0-1. The Levenberg-Marquardt
% training algorithm was modified to accommodate this peculiar training requirement.
% It can be found in file 'trainlm_posw3.m'.
%-----
% S.A.B.           NASA          10/10/2001
%-----

load 181222f1_adas_aol.bin.mat;           % Oct. 6, 1997 inbound track
load 003853ad.cal_new;                   % Load ADAS calibration file
Chloro    = Chloro.*3;                   % Convert to mg/m^3

s       = size(Chloro);                  % Get size of file
Radiance = Radiance ./ (X003853ad(:,3)*ones(1,s(2))); % Recover raw counts
P       = Radiance([11 35 70 85 119],:); % Reduce to 5 channels centered at
                                         % 412, 443, 490, 510, and 555 nm

M       = mean(P);                      % Used to normalize data with zero mean
SD      = std(P);                      % and unity std.

I       = ones(5,1);                    % Vectorize for direct calculation
M       = I*M;
SD      = I*SD;

P       = (P - M). / SD;                % Normalize spectra as described above
P       = P.*5;                         % Increase voltage on all channels to
                                         % compensate for passive circuit

%-----
half   = floor(s(2)/2);                 % Calc. half of data
ch     = Chloro./20;                   % Divide chlorophyll by 20
index  = randperm(s(2));                % Create random index the size of data
rch    = ch(index);                   % Create shuffled data sets
rP     = P(:,index);

%-----
% This part trains on 3000 pts. from the 'shuffled' deck and
% then does a verification on another (different) 3000 pts. from
% same deck. When verification error stops decreases or Threshold is met,
% training stops.
%-----

Threshold = .03;                       % Stop training at this threshold
while(1)
    index = randperm(s(2));            % Create random index the size of data
```

```

rch    = ch(index);                                % Create shuffled data sets
rP     = P(:,index);

for i=1:5                                         % Loop 5 times before shuffling deck again
    % Initialize weights and biases
    [W1,B1,W2,B2] = initff(P,2,'diodesig6',ch,'purelin');
    OldError = 1234567e45;

        % Make a training run of 300 interations
    [W1,B1,W2,B2,TE,TR] = trainlm_posw3(W1,B1,'diodesig6',W2,B2,'purelin',
                                          rP(:,1:half),rch(1:half),[10 300 Threshold]);
        % Test training coefficients with other
        % half of data. Remember to divide weights
        % and biases of input layer by 6 to match
        % the 6 input, summation network of
        % circuit3. Also, remember to divide the
        % output by 3 to compensate for the second
        % (output layer) summation network of
        % circuit3.

    Test  = simuff(rP(:,half+1:s(2))./6,W1,B1./6,'diodesig6',W2,B2,'purelin')./3;
    Diff  = Test-rch(half+1:s(2));

    Error = sumsqr(Diff);                           % Calc. sum squared error on test set
    m = max(max(abs(W1)));                         % Find the MAX ABS of matrix W1
    % Print max and current error
    fprintf('\nMax W1 = %f      SSE = %f\n\n',m>Error);

        % We are looking for a maximum, positive
        % weight (in W1) <= 1 because circuit3 only
        % represents positive weights up to 1.0.
        % When a training set meets both this
        % criteria and a sum squared error <=
        % our threshold, then we stop training.

    if ( (m <= 1.0) & (Error <= Threshold) )
        break;
    end
end
                                                % Need to repeat this to exit our for
                                                % loop

if ( (m <= 1.0) & (Error <= Threshold) )
    break;
end

end

```

The following Matlab code is a modified library routine that comes with the Neural Network Toolbox for Matlab. This function is actually a setup routine for calling specific neural network functions used for training. I simply replaced the functions used to train 2 layer neural networks with a custom function titled ‘tlm2_posw3.m’.

```

function [a,b,c,d,e,f,g,h] = trainlm_posw3(i,j,k,l,m,n,o,p,q,r,s,t)
%TRAINLM Train feed-forward network with Levenberg-Marquardt.
%
% TRAINLM can be called with 1, 2, or 3 sets of weights
% and biases to train up to 3 layer feed-forward networks.
%
% [W1,B1,W2,B2,...,TE,TR] = TRAINLM(W1,B1,F1,W2,B2,F2,...,P,T,TP)
%   Wi - SixR weight matrix for the ith layer.
%   Bi - S1x1 bias vector for the ith layer.
%   Fi - Transfer function (string) for the ith layer.
%   P - RxQ matrix of input vectors.
%   T - SxQ matrix of target vectors.
%   TP - Training parameters (optional).
% Returns new weights and biases and
%   Wi - new weights.
%   Bi - new biases.
%   TE - the actual number of epochs trained.
%   TR - training record: [row of errors]
%
% Training parameters are:
%   TP(1) - Epochs between updating display, default = 25.
%   TP(2) - Maximum number of epochs to train, default = 1000.
%   TP(3) - Sum-squared error goal, default = 0.02.
%   TP(4) - Minimum gradient, default = 0.0001.
%   TP(5) - Initial value for MU, default = 0.001.
%   TP(6) - Multiplier for increasing MU, default = 10.
%   TP(7) - Multiplier for decreasing MU, default = 0.1.
%   TP(8) - Maximum value for MU, default = 1e10.
% Missing parameters and NaN's are replaced with defaults.
%
% See also NNTRAIN, BACKPROP, INITFF, SIMFF, TRAINBP, TRAINBPX.

% Mark Beale, 12-15-93
% Copyright (c) 1992-97 by The MathWorks, Inc.
% $Revision: 1.3 $ $Date: 1997/05/14 22:30:19 $

if all([5 6 8 9 11 12] ~= nargin),error('Wrong number of input arguments'),end

if nargin == 5
    [a,b,c,d] = tlml(i,j,k,l,m);
elseif nargin == 6
    [a,b,c,d] = tlml(i,j,k,l,m,n);
elseif nargin == 8
    [a,b,c,d,e,f] = tlm2_posw3(i,j,k,l,m,n,o,p);
elseif nargin == 9
    [a,b,c,d,e,f] = tlm2_posw3(i,j,k,l,m,n,o,p,q);
elseif nargin == 11
    [a,b,c,d,e,f,g,h] = tlm3(i,j,k,l,m,n,o,p,q,r,s);
elseif nargin == 12
    [a,b,c,d,e,f,g,h] = tlm3(i,j,k,l,m,n,o,p,q,r,s,t);
end

```

The following Matlab code is a modified library routine that comes with the Neural Network Toolbox for Matlab. This function is used to train a 2 layer neural network. Changes to this library function include the following:

1. Weights of the input layer are kept positive. This models our circuits voltage dividers which have one end tied to ground.
2. Weights of the output layer are fixed at a value of 1.0. This models our circuit with output weights set to unity.
3. Input layer weights and biases are divided by 6 . This models our circuit with 6 inputs arranged in a divide by 6 configuration.
4. Output layer response is divided by 3. This models our circuit with 3 inputs to the output layer arranged in a divide by 3 configuration.

```

function [w1,b1,w2,b2,i,tr] = tlm2_posw3(w1,b1,f1,w2,b2,f2,p,t,tp)
%TLM2 Train 2-layer feed-forward network w/Levenberg-Marquardt.
%
%      [W1,B1,W2,B2,TE,TR] = TLM2(W1,B1,'F1',W2,B2,'F2',P,T)
%      Wi - Weight matrix of ith layer.
%      Bi - Bias vector of ith layer.
%      F - Transfer function (string) of ith layer.
%      P - RxQ matrix of input vectors.
%      T - S2xQ matrix of target vectors.
%      TP - Training parameters (optional).
%
% Returns:
%      Wi - new weights.
%      Bi - new biases.
%      TE - the actual number of epochs trained.
%      TR - training record: [row of errors]
%
% Training parameters are:
%      TP(1) - Epochs between updating display, default = 25.
%      TP(2) - Maximum number of epochs to train, default = 1000.
%      TP(3) - Sum-squared error goal, default = 0.02.
%      TP(4) - Minimum gradient, default = 1e-6.
%      TP(5) - Initial value for MU, default = 0.001.
%      TP(6) - Multiplier for increasing MU, default = 10.
%      TP(7) - Multiplier for decreasing MU, default = 0.1.
%      TP(8) - Maximum value for MU, default = 1e10.
%
% Missing parameters and NaN's are replaced with defaults.

% Mark Beale, 12-15-93, MB
% Copyright (c) 1992-97 by The MathWorks, Inc.
% $Revision: 1.3 $ $Date: 1997/05/14 22:27:04 $

if nargin < 8,error('Not enough arguments.'),end

% TRAINING PARAMETERS
if nargin == 8, tp = []; end
tp = mndef(tp,[25 1000 0.02 1e-6 0.001 10 0.1 1e10]);
df = tp(1);
me = tp(2);
eg = tp(3);
grad_min = tp(4);
mu_init = tp(5);
mu_inc = tp(6);
mu_dec = tp(7);
mu_max = tp(8);
df1 = feval(f1,'delta');
df2 = feval(f2,'delta');

% DEFINE SIZES
[s1,r] = size(w1);
[s2,s1] = size(w2);
w1_ind = [1:(s1*r)];
b1_ind = [1:s1] + w1_ind(length(w1_ind));

```

```

w2_ind = [1:(s1*s2)] + b1_ind(length(b1_ind));
b2_ind = [1:s2] + w2_ind(length(w2_ind));
ii = eye(b2_ind(length(b2_ind)));
dw1 = w1; db1 = b1;
dw2 = w2; db2 = b2;
ext_p = nncpyi(p,s2);

% PRESENTATION PHASE
[a1,a2] = simuff(p./6,w1,b1./6,f1,w2,b2,f2);
a1 = a1./3; % SAB 10/10/01
a2 = a2./3; % This is hardwired for a
% 5x2x1 passive network. We
% divide by 6 in input layer
% to simulate attenuation
% of 6 voltage dividers of passive
% circuit (circuit 3). The output
% is divided by 3 to simulate
% the attenuation of 3 voltage
% dividers on output of circuit
e = t-a2;
SSE = sumsqr(e);

% TRAINING RECORD
tr = zeros(1,me+1);
tr(1) = SSE;

% PLOTTING FLAG
plottype = (r==1) & (s2==1);

% PLOTTING
newplot;
message = sprintf('TRAINLM: %%g/%g epochs, mu = %%g, SSE = %%g.\n',me);
fprintf(message,0,mu_init,SSE)
if plottype
    h = plotfa(p,t,p,a2);
else
    h = ploterr(tr(1),eg);
end

mu = mu_init;
for i=1:me

    % CHECK PHASE
    if SSE < eg, i=i-1; break, end

    % FIND JACOBIAN

    ext_a1 = nncpyi(a1,s2);
    d2 = feval(df2,a2);
    ext_d2 = -nncpyd(d2);
    ext_d1 = feval(df1,ext_a1,ext_d2,w2);
    j1 = learnlm(ext_p,ext_d1);
    j2 = learnlm(ext_a1,ext_d2);
    j = [j1, ext_d1', j2, ext_d2'];

    % CHECK MAGNITUDE OF GRADIENT
    je = j' * e(:);
    grad = norm(je);
    if grad < grad_min, i=i-1; break, end

    % INNER LOOP, INCREASE MU UNTIL THE ERRORS ARE REDUCED
    jj = j'*j;

    while (mu <= mu_max)
        dx = -(jj+ii*mu) \ je;
        dw1(:) = dx(w1_ind); db1 = dx(b1_ind);
        dw2(:) = dx(w2_ind); db2 = dx(b2_ind);
        new_w1 = w1 + dw1; new_b1 = b1 + db1;
        new_w2 = w2 + dw2; new_b2 = b2 + db2;

        I = find(new_w1<0); % SAB 10/10/01...only want positive weights
        new_w1(I) = 0;

```

```

I = find(new_w2<0); % SAB 10/10/01...only want positive weights
new_w2(I) = 0;

new_w2 = [ 1 1 ]; % SAB 10/10/01...output layer weights remain at 1

% EVALUATE NEW NETWORK
[a1,a2] = simuff(p./6,new_w1,new_b1./6,f1,new_w2,new_b2,f2);
a1 = a1./3; % SAB 10/10/01...the same logic is used here
a2 = a2./3; % as described above in the presentation phase
new_e = t-a2;
new_SSE = sumsqr(new_e);

if (new_SSE < SSE), break, end
mu = mu * mu_inc;
end
if (mu > mu_max), i = i-1; break, end
mu = mu * mu_dec;

% UPDATE NETWORK
w1 = new_w1; b1 = new_b1;
w2 = new_w2; b2 = new_b2;
e = new_e; SSE = new_SSE;

% TRAINING RECORD
tr(i+1) = SSE;

% PLOTTING
if rem(i,df) == 0
    fprintf(message,i,mu,SSE)
    if plottype
        delete(h); h = plot(p,a2,'m'); drawnow;
    else
        h = ploterr(tr(1:(i+1)),eg,h);
    end
end
end

% TRAINING RECORD
tr = tr(1:(i+1));

% PLOTTING
if rem(i,df) ~= 0
    fprintf(message,i,mu,SSE)
    if plottype
        delete(h);
        plot(p,a2,'m');
        drawnow;
    else
        ploterr(tr,eg,h);
    end
end
end

% WARNINGS
if SSE > eg
    disp(' ')
    if (mu > mu_max)
        disp('TRAINLM: Error gradient is too small to continue learning.')
    else
        disp('TRAINLM: Network error did not reach the error goal.')
    end
    disp(' Further training may be necessary, or try different')
    disp(' initial weights and biases and/or more hidden neurons.')
    disp(' ')
end

```

The following Matlab code is a modified library routine that comes with the Neural Network Toolbox for Matlab. This function is a proxy for the Matlab neural network transfer function titled ‘tansig.m’. Where ‘tansig’ used a simple equation to represent a transfer function, we used 3 polynomials connected piecewise. Data for these polynomials was derived from measured data found in the file titled ‘p1n914_measure.mat’. The basis of this file is a text file described in ‘Appendix C – Tables’ titled ‘p1n914_measure.txt’.

```

function a = diodesig3(n,b)
%STEVESIG Variation of hyperbolic tangent sigmoid transfer function.
%
% STEVESIG(N)
%   N - SxQ Matrix of net input (column) vectors.
%   Returns the values of N squashed between -1 an 1.
%
% EXAMPLE: n = -10:0.1:10;
%           a = stevesig(n);
%           plot(n,a)
%
% STEVESIG(Z,B) ...Used when Batching.
%   Z - SxQ Matrix of weighted input (column) vectors.
%   B - Sx1 Bias (column) vector.
%   Returns the squashed net input values found by adding
%   B to each column of Z.
%
% STEVESIG('delta') returns name of delta function.
% STEVESIG('init') returns name of initialization function.
% STEVESIG('name') returns full name of this transfer function.
% STEVESIG('output') returns output range of this function.
%
% See also NNTRANS, BACKPROP, NWTAN, LOGSIG.

% Mark Beale, 1-31-92
% Revised 12-15-93, MB
% Copyright (c) 1992-97 by The MathWorks, Inc.
% $Revision: 1.3 $ $Date: 1997/05/14 22:23:09 $
%
% This version is a modification of 'tansig' made by:
% Steven A. Bailey      NASA      5/30/2001
% This function is an approximation to one diode (ln914)

if nargin < 1, error('Not enough arguments.'); end

if isstr(n)
    if strcmp(lower(n),'delta')
        a = 'deltadiode3';
    elseif strcmp(lower(n),'init')
        a = 'nwtan';
    elseif strcmp(lower(n),'name')
        a = 'Hyperbolic Tangent Sigmoid';
    elseif strcmp(lower(n),'output')
        a = [-1 1];
    else
        error('Unrecognized property.')
    end
else
    if nargin==2
        [nr,nc] = size(n);
        n = n + b*ones(1,nc);
    end
end

Ibottom = find(n<-1.5);
Imiddle = find(n>=-1.5 & n<=1.5);
Itop = find(n>1.5);

load p1n914_measure.mat;      % Load in coefficients

```

```
n(Ibottom)      = polyval(Pbottom,n(Ibottom));
n(Imiddle)      = polyval(Pmiddle,n(Imiddle));
n(Itop)          = polyval(Ptop,n(Itop));

a = n;
i = find(~finite(a));
a(i) = sign(n(i));
end
```

The following Matlab code is a modified library routine that comes with the Neural Network Toolbox for Matlab. This function is a proxy for the Matlab neural network transfer function titled ‘deltatan.m’. Where ‘deltatan’ used a simple equation to represent the derivative at any given point on the ‘tansig’ transfer function, we compute the derivative of 3 polynomials connected piecewise. Data for these polynomials was derived from measured data found in the file titled ‘p1n914_measure.mat’. The basis of this file is a text file described in ‘Appendix C – Tables’ titled ‘p1n914_measure.txt’.

```

function d = deltadiode3(a,d,w)
%DELTALOG Delta function for LOGSIG neurons.
%
% DELTALIN(A)
%   A - S1xQ matrix of output vectors.
% Returns the S1xQ matrix of derivatives of the output vectors
%   with respect to the net input of the PURELIN transfer function.
%
% DELTALOG(A,E)
%   E - S1xQ matrix of associated errors
% Returns an S1xQ matrix of derivatives of error for an output layer.
%
% DELTALOG(A,D,W)
%   D - S2xQ matrix of next layer delta vectors
%   W - S2xS1 weight matrix between layers.
% Returns an S1xQ matrix of derivatives of error for a hidden layer.
%
% See also NNTRANS, BACKPROP, LOGSIG, DELTALIN, DELTATAN

% Mark Beale, 1-31-92
% Revised 12-15-93, MB
% Copyright (c) 1992-97 by The MathWorks, Inc.
% $Revision: 1.3 $ $Date: 1997/05/14 21:56:04 $

if nargin < 1,error('Not enough input arguments'),end

% We need to find independant variable first, given the dependant variable(s) 'a'

load p1n914_measure.mat;      % Load in coefficients

x = polyval(Pinverse, a);

Ibottom = find(x<-1.5);
Imiddle = find(x>=-1.5 & x<=1.5);
Itop    = find(x>1.5);

a(Ibottom)    = polyval(Pbottom_deriv,x(Ibottom));
a(Imiddle)    = polyval(Pmiddle_deriv,x(Imiddle));
a(Itop)       = polyval(Ptop_deriv,x(Itop));

if nargin == 1
    d = a;
elseif nargin == 2
    d = a.*d;
else
    d = a.*(w'*d);
end

```

The following Matlab code is a modified library routine that comes with the Neural Network Toolbox for Matlab. This function is a proxy for the Matlab neural network transfer function titled ‘tansig.m’. Where ‘tansig’ used a simple equation to represent a transfer function, we used 3 polynomials connected piecewise. Data for these polynomials was derived from measured data found in the file titled ‘d1n914_circuit3.mat’. The basis of this file is a text file described in ‘Appendix C – Tables’ titled ‘d1n914_circuit3.txt’.

```

function a = diodesig6(n,b)
%STEVESIG Variation of hyperbolic tangent sigmoid transfer function.
%
% STEVESIG(N)
%   N - SxQ Matrix of net input (column) vectors.
%   Returns the values of N squashed between -1 an 1.
%
% EXAMPLE: n = -10:0.1:10;
%           a = stevesig(n);
%           plot(n,a)
%
% STEVESIG(Z,B) ...Used when Batching.
%   Z - SxQ Matrix of weighted input (column) vectors.
%   B - Sx1 Bias (column) vector.
%   Returns the squashed net input values found by adding
%   B to each column of Z.
%
% STEVESIG('delta') returns name of delta function.
% STEVESIG('init') returns name of initialization function.
% STEVESIG('name') returns full name of this transfer function.
% STEVESIG('output') returns output range of this function.
%
% See also NNTRANS, BACKPROP, NWTAN, LOGSIG.

% Mark Beale, 1-31-92
% Revised 12-15-93, MB
% Copyright (c) 1992-97 by The MathWorks, Inc.
% $Revision: 1.3 $ $Date: 1997/05/14 22:23:09 $
%
% This version is a modification of 'tansig' made by:
% Steven A. Bailey      NASA      10/10/2001
% This function is an approximation to one diode (ln914)
% taken from real data by measuring circuit3

if nargin < 1, error('Not enough arguments.'); end

if isstr(n)
    if strcmp(lower(n),'delta')
        a = 'deltadiode6';
    elseif strcmp(lower(n),'init')
        a = 'nwtan';
    elseif strcmp(lower(n),'name')
        a = 'Hyperbolic Tangent Sigmoid';
    elseif strcmp(lower(n),'output')
        a = [-1 1];
    else
        error('Unrecognized property.')
    end
else
    if nargin==2
        [nr,nc] = size(n);
        n = n + b*ones(1,nc);
    end
    Ibottom = find(n<-1.5);
    Imiddle = find(n>=-1.5 & n<=1.5);
    Itop    = find(n>1.5);

    load d1n914_circuit3.mat;    % Load in coefficients

```

```
n(Ibottom)      = polyval(Pbottom,n(Ibottom));
n(Imiddle)     = polyval(Pmiddle,n(Imiddle));
n(Itop)         = polyval(Ptop,n(Itop));

a = n;
i = find(~finite(a));
a(i) = sign(n(i));
end
```

The following Matlab code is a modified library routine that comes with the Neural Network Toolbox for Matlab. This function is a proxy for the Matlab neural network transfer function titled ‘deltatan.m’. Where ‘deltatan’ used a simple equation to represent the derivative at any given point on the ‘tansig’ transfer function, we compute the derivative of 3 polynomials connected piecewise. Data for these polynomials was derived from measured data found in the file titled ‘d1n914_circuit3.mat’. The basis of this file is a text file described in ‘Appendix C – Tables’ titled ‘d1n914_circuit3.txt’.

```

function d = deltadiode6(a,d,w)
%DELTALOG Delta function for LOGSIG neurons.
%
% DELTALIN(A)
%   A - S1xQ matrix of output vectors.
% Returns the S1xQ matrix of derivatives of the output vectors
%   with respect to the net input of the PURELIN transfer function.
%
% DELTALOG(A,E)
%   E - S1xQ matrix of associated errors
% Returns an S1xQ matrix of derivatives of error for an output layer.
%
% DELTALOG(A,D,W)
%   D - S2xQ matrix of next layer delta vectors
%   W - S2xS1 weight matrix between layers.
% Returns an S1xQ matrix of derivatives of error for a hidden layer.
%
% See also NNTRANS, BACKPROP, LOGSIG, DELTALIN, DELTATAN

% Mark Beale, 1-31-92
% Revised 12-15-93, MB
% Copyright (c) 1992-97 by The MathWorks, Inc.
% $Revision: 1.3 $ $Date: 1997/05/14 21:56:04 $

if nargin < 1,error('Not enough input arguments'),end

% We need to find independant variable first, given the dependant variable(s) 'a'

load dln914_circuit3.mat;      % Load in coefficients

x = polyval(Pinverse, a);

Ibottom = find(x<-1.5);
Imiddle = find(x>=-1.5 & x<=1.5);
Itop    = find(x>1.5);

a(Ibottom)    = polyval(Pbottom_deriv,x(Ibottom));
a(Imiddle)    = polyval(Pmiddle_deriv,x(Imiddle));
a(Itop)       = polyval(Ptop_deriv,x(Itop));

if nargin == 1
    d = a;
elseif nargin == 2
    d = a.*d;
else
    d = a.*(w'*d);
end

```

The following Matlab code is used to both convert and parse Sun generated ADAS data into a format compatible with Matlab. The sole variable used for this study is ‘radiance’ which comes out as a vector of 256 channels spanning 398.0 to 737.2 nm with channel centers 1.3 nm apart.

```
%-----  
% This function reads a Sun produced, binary file of ADAS data  
% from Oct. 6, 1997 and creates a Matlab file with all variables  
% parsed and placed in separate variables.  
%-----  
% SAB      NASA      3/20/01  
%-----  
  
function parsedata( filename )  
  
M      = sun2pc(filename);          % Convert to PC format  
s      = size(M);  
  
Time   = M(1121:1120:s(1))';     % Parse variables  
Range  = M(1122:1120:s(1))';  
Lat    = M(1123:1120:s(1))';  
Lon    = M(1124:1120:s(1))';  
Pitch  = M(1125:1120:s(1))';  
Roll   = M(1126:1120:s(1))';  
TX1    = M(1127:1120:s(1))';  
TX2    = M(1128:1120:s(1))';  
RX1    = M(1129:1120:s(1))';  
RX2    = M(1130:1120:s(1))';  
SST    = M(1131:1120:s(1))';  
RUV    = M(1132:1120:s(1))';  
RG     = M(1133:1120:s(1))';  
FR     = M(1134:1120:s(1))';  
Pub    = M(1135:1120:s(1))';  
Peb    = M(1136:1120:s(1))';  
Chloro = M(1137:1120:s(1))';  
PubPeb = M(1144:1120:s(1))';  
  
s      = size(Time);  
  
Irradiance = zeros(6,s(2));        % Make space for uplooking radiances  
Radiance   = zeros(256,s(2));       % Make space for ADAS radiances  
cnt        = 1138;                  % Start of Irradiance  
  
for i=1:s(2)                      % Pull out Irradiance & Radiance  
    Irradiance(:,i) = M(cnt:cnt+5);  
    Radiance(:,i)  = M(cnt+7:cnt+7+255);  
    cnt = cnt + 1120;  
end  
  
command = sprintf('save %s.mat',filename);  
clear i s filename cnt M;  
eval(command);
```

The following Matlab code is called from the function above (parsedata.m). Its purpose is to rearrange Sun packed floating point numbers into an Intel compatible arrangement.

```
%-----  
% This function reads in a Sun produced binary file and produces  
% a PC/Matlab readable array. Sun produces Big Endian and  
% PC produces Little Endian. All we need to do is invert or  
% mirror Sun bits to PC bits. IE. - bit 32 becomes bit 1 and  
% vice versa. The only way I could figure out how to do this  
% is to read in each byte one at a time, and then write them  
% out in reversed order 4 bytes at a time. I then read them  
% back in and they convert directly to floats.  
%-----  
% SAB          NASA          12/11/98  
%-----  
  
function M = sun2pc( filename )  
  
fid = fopen( filename, 'r' );           % Read in 1 char at a time  
[A, Count] = fread( fid, 'uchar' );  
fclose( fid );  
  
fid = fopen( 'junk.bin', 'w' );           % Write them back out reversed  
for (i=4:4:Count)  
    fwrite( fid, A(i:-1:i-3), 'uchar' );  
end  
fclose( fid );  
  
fid = fopen( 'junk.bin', 'r' );           % Read back in and convert to float  
[M, Count] = fread( fid, 'float' );  
fclose( fid );
```

Appendix B – Microchip Source Code

The following Microchip assembler code is used to implement our custom designed DAC board which drives either the active or passive AANN boards.

```

;-----  

; Dac2.asm      This version is the second in a series which is used  

;                to load and drive a 5 channel DAC with stored values  

;                based ADAS spectral data recorded on 10/06/97. This is  

;                the default state. This version also waits (in the  

;                foreground) for any activity on the serial port. It is  

;                here both commands and data can arrive from an external  

;                PC.  

;-----  

; Steven A. Bailey    NASA        8/10/01  

;-----  

list          p=16f874, r=dec      ; Define processor and 'dec' numbers  

; as default  

include <p16f874.inc>           ; Processor specific defines  

include <pxxmacs.inc>          ; Parallax 'like' macros  

; These are the configuartion bits found  

; in 'p16c65a.inc'.  

__config      _HS_OSC & _WDT_OFF & _PWRTE_ON & _CP_OFF & _BODEN_ON & _LVE_OFF  

;-----  

; Program defines  

;-----  

#define SPBRG_VAL     129          ; Baud Rate = high  

; SPBRG_VAL =      FOSC  

; ----- - 1  

; (BAUDRATE*16)  

;  

; For 9.6 Kbaud:  

; SPBRG_VAL = 20,000,000  

; ----- - 1 = 129  

; (9600*16)  

#define LOW_TIME      0x15          ; Low byte of startup time  

#define HIGH_TIME     0xfe          ; High byte of startup time  

; Since this is a count-up timer  

; 10000h - ff06h = 00fah or 250  

; Since a timer tic is 200 ns, then  

; 250 x 200 ns = 50 usecs between  

; interrupts. I added 9 to calculated  

; value because it takes 9 cycles to  

; get to handler after interrupt. This  

; was determined using simulator  

#define DTOA_A0        PORTA,0       ; DTOA address line 0  

#define DTOA_A1        PORTA,1       ; DTOA address line 1  

#define DTOA1_WR       PORTB,0       ; DTOA1 WR line  

#define DTOA1_LAC      PORTB,1       ; DTOA1 LAC line  

#define DTOA1_CMB      PORTB,2       ; DTOA1 CMB line  

#define DTOA1_CLB      PORTB,3       ; DTOA1 CLB line  

#define DTOA2_WR       PORTB,4       ; DTOA2 WR line  

#define DTOA2_LAC      PORTB,5       ; DTOA2 LAC line  

#define DTOA2_CMB      PORTB,6       ; DTOA2 CMB line  

#define DTOA2_CLB      PORTB,7       ; DTOA2 CLB line  

#define DTOA_D0        PORTD,0       ; DTOA data line 0  

#define DTOA_D1        PORTD,1       ; DTOA data line 1

```

```

#define DTOA_D2      PORTD,2      ; DTOA data line 2
#define DTOA_D3      PORTD,3      ; DTOA data line 3
#define DTOA_D4      PORTD,4      ; DTOA data line 4
#define DTOA_D5      PORTD,5      ; DTOA data line 5
#define DTOA_D6      PORTD,6      ; DTOA data line 6
#define DTOA_D7      PORTD,7      ; DTOA data line 7

;-----
; Variables in RAM
;-----
; A more consistant way to define RAM is to use the 'org' statement
; followed by variables using the 'res' directive. However, the RAM
; window of 'MPLAM' does not display variables defined this way...hence
; we are using the 'equ' directive followed by the address in the
; register map they are located.
;-----
;          org  0x20          ; Start of RAM in 16c65a
;
;var1      res   1           ; Examples using 'res'
;var2      res   1
;-----

dtoal_low    equ    0x20      ; DTOA channel 1 low byte
dtoal_high   equ    0x21      ; DTOA channel 1 high byte
dtoa2_low    equ    0x22      ; DTOA channel 2 low byte
dtoa2_high   equ    0x23      ; DTOA channel 2 high byte
dtoa3_low    equ    0x24      ; DTOA channel 3 low byte
dtoa3_high   equ    0x25      ; DTOA channel 3 high byte
dtoa4_low    equ    0x26      ; DTOA channel 4 low byte
dtoa4_high   equ    0x27      ; DTOA channel 4 high byte
dtoa5_low    equ    0x28      ; DTOA channel 5 low byte
dtoa5_high   equ    0x29      ; DTOA channel 5 high byte
w_copy       equ    0x2a      ; Used for stack manipulation
s_copy       equ    0x2b
flash_low    equ    0x2c      ; Low byte of FLASH address
flash_high   equ    0x2d      ; High byte of FLASH address
data_low     equ    0x2e      ; Data low byte
data_high    equ    0x2f      ; Data high byte
dszie_low    equ    0x30      ; Data size low byte
dszie_high   equ    0x31      ; Data size high byte
arbit_flag   equ    0x32      ; Arbitration flag
temp         equ    0x33      ; Temp variable
rcv_byte     equ    0x34      ; Receive byte
xmt_byte     equ    0x35      ; Transmit byte

;-----
; On startup, the PIC looks at addequs 0 for its first
; instruction. Since the interrupt handler begins at
; addequs 4, we'll just jump over it to get to the
; startup routine.
;-----
;          org  0          ; equiet vector location
;          jmp  start        ; Beginning of main program.

;-----
; Next is the interrupt handler, which must begin at
; addequs 4. This handler copies equtoequ w and the status
; register. Because a normal "mov w,fr" alters the z bit of
; the status register, this routine uses "mov w,<>fr," which
; does not. The routine actually swaps the byte twice,
; equulting in the correct value being written to w without
; affecting the z bit. This routine is setup to be called
; every msec.
;-----
;          org  4          ; Interrupt vector location
;          jmp  handler      ; Go to timer1 interrupt routine

;-----
; Here's the startup routine and the main program loop.

```

```

; In the line that initializes "intcon," bit 7 is GIE and bit 5
; is RTIE. Writing 1s to these enables interrupts generally (GIE)
; and the RTCC interrupt specifically (RTIE).
;-----

start
    call    initialize           ; Initialize CPU
mainloop
    snb    PIR1,RCIF           ; Skip if Receiver not ready
    call    parse
    jmp    mainloop

;-----
; Here's our interrupt timer routine.
; In the line that initializes INTCON, bit 7 is GIE and bit 6
; is PEIE. At the moment, the interrupt handler is being called
; every 10 ms or at a 100 Hz rate.
; From call to return, this function takes 44975 cycles (worst case) or
; 8.99 msecs. This worst case occurs at most, once every 200 msecs...or
; once every 20 calls to this interrupt handler.
; Average number of cycles used in this function when not writing data
; is 133 cycles or 26.6 usecs.
;-----


handler
    clrb   STATUS,RP0           ; Switch to Memory Bank 0
    clrb   PIR1,TMR1IF          ; Clear TMR1 interrupt flag

    mov_ff w_copy,w             ; Make a copy of w.
    mov_ff s_copy,STATUS         ; Make a copy of status.
    mov_fl TMR1L,LOW_TIME       ; Load low byte of TMR1
    mov_fl TMR1H,HIGH_TIME      ; Load high byte of TMR1

    cje_fl arbit_flag,1,cont    ; If arbit_flag == 1, then
                                ; foreground process is busy
                                ; updating FLASH data

    call    readflash           ; Read in 1st 14bit word
    mov_ff dtoal_low,data_low
    mov_ff dtoal_high,data_high

    call    readflash           ; Read in 2nd 14bit word
    mov_ff dtoa2_low,data_low
    mov_ff dtoa2_high,data_high

    call    readflash           ; Read in 3rd 14bit word
    mov_ff dtoa3_low,data_low
    mov_ff dtoa3_high,data_high

    call    readflash           ; Read in 4th 14bit word
    mov_ff dtoa4_low,data_low
    mov_ff dtoa4_high,data_high

    call    readflash           ; Read in 5th 14bit word
    mov_ff dtoa5_low,data_low
    mov_ff dtoa5_high,data_high

    call    dtoa                  ; Call dtoa routine to set 5 voltages

cont
    mov_ff STATUS,s_copy        ; restore status register
    restw  w_copy               ; restore w without affecting STATUS bits
    reti

;-----
; Set all 5 voltages with this routine using variables gathered
; from main routine.
;-----


dtoa
    setb   DTOA1_LAC            ; Latch held high until ready to output all channels

```

```

setb DTOA_A0           ; Set address line to register B
clr b DTOA_A1
mov_ff PORTD,dtoal_low ; Load Port D with low byte of channel 1
clr b DTOA1_CLB
setb DTOA1_CMB
clr b DTOA1_WR
setb DTOA1_WR
mov_ff PORTD,dtoal_high; Load Port D with high byte of channel 1
setb DTOA1_CLB
clr b DTOA1_CMB
clr b DTOA1_WR
setb DTOA1_WR           ; Latch MS byte into register B

clr b DTOA_A0           ; Set address line to register C
setb DTOA_A1
mov_ff PORTD,dtoa2_low ; Load Port D with low byte of channel 2
clr b DTOA1_CLB
setb DTOA1_CMB
clr b DTOA1_WR
setb DTOA1_WR
mov_ff PORTD,dtoa2_high; Load Port D with high byte of channel 2
setb DTOA1_CLB
clr b DTOA1_CMB
clr b DTOA1_WR
setb DTOA1_WR           ; Latch MS byte into register C

setb DTOA_A0           ; Set address line to register D
setb DTOA_A1
mov_ff PORTD,dtoa3_low ; Load Port D with low byte of channel 3
clr b DTOA1_CLB
setb DTOA1_CMB
clr b DTOA1_WR
setb DTOA1_WR
mov_ff PORTD,dtoa3_high; Load Port D with high byte of channel 3
setb DTOA1_CLB
clr b DTOA1_CMB
clr b DTOA1_WR
setb DTOA1_WR           ; Latch MS byte into register D

setb DTOA2_LAC          ; Latch held high until ready to output all channels

setb DTOA_A0           ; Set address line to register B
clr b DTOA_A1
mov_ff PORTD,dtoa4_low ; Load Port D with low byte of channel 4
clr b DTOA2_CLB
setb DTOA2_CMB
clr b DTOA2_WR
setb DTOA2_WR
mov_ff PORTD,dtoa4_high; Load Port D with high byte of channel 4
setb DTOA2_CLB
clr b DTOA2_CMB
clr b DTOA2_WR
setb DTOA2_WR           ; Latch MS byte into register B

clr b DTOA_A0           ; Set address line to register C
setb DTOA_A1
mov_ff PORTD,dtoa5_low ; Load Port D with low byte of channel 5
clr b DTOA2_CLB
setb DTOA2_CMB
clr b DTOA2_WR
setb DTOA2_WR
mov_ff PORTD,dtoa5_high; Load Port D with high byte of channel 5
setb DTOA2_CLB
clr b DTOA2_CMB
clr b DTOA2_WR
setb DTOA2_WR           ; Latch MS byte into register C

mov_f1 PORTB,00010001b   ; Transfer data to all 5 DACs at once
; with the following mapping:
;-----

```

```

;      DTOA2      |      DTOA1
;-----|-----|
; CLB CMB LAC WR CLB CMB LAC WR
;-----|-----|
; 0  0  0  1  0  0  0  1
;-----|-----|


return

;-----
; Initialize serial port and variables
; From call to return, this function takes 70 cycles or 14 usecs.
;-----


initialize
    setb STATUS,RP0           ; Switch to Memory Bank 1

    mov_f1 TRISA, 00000000b   ; Make ra0-7 outputs
    mov_f1 TRISB, 00000000b   ; Make rb0-7 outputs
    mov_f1 TRISC, 10000010b   ; Make rc1 & rc7 pins inputs
    mov_f1 TRISD, 00000000b   ; Make rd0-7 outputs
    mov_f1 TRISE, 00000000b   ; Make re0-7 outputs
    mov_f1 ADCON1,00000110b   ; Disable A/D convertors

    mov_f1 SPBRG,SPBRG_VAL   ; Set baud rate generator
    mov_f1 TXSTA,00100100b   ; Trans enable, 8-bits, async, baud low
    mov_f1 PIE1,00000001b     ; Enable TMR1 interrupt on overflow

    clr b STATUS,RP0          ; Switch to Memory Bank 0

    mov_f1 flash_low,0         ; Setup for FLASH memory read
    mov_f1 flash_high,8
    mov_f1 dsize_low,0
    mov_f1 dsize_high,9        ; Must initialize so > flash
    call readflash
    mov_ff dsize_low,data_low  ; Get size of data from 1st word
    mov_ff dsize_high,data_high

    clr arbit_flag             ; Clear arbitration flag...used for
                                ; arbitration between foreground and
                                ; background processes

    setb DTOA1_CLB            ; Turn U1 (MAX527) off
    setb DTOA1_CMB
    clrb DTOA1_WR
    setb DTOA1_LAC

    setb DTOA2_CLB            ; Turn U2 (MAX527) off
    setb DTOA2_CMB
    clrb DTOA2_WR
    setb DTOA2_LAC

    mov_f1 RCSTA,10010000b    ; Serial port enable
    mov_ff rcv_byte,RCREG     ; Clear out receive buffer

    mov_f1 TMR1L,LOW_TIME     ; Load low byte of TMR1
    mov_f1 TMR1H,HIGH_TIME    ; Load high byte of TMR1
    mov_f1 T1CON,00000001b     ; Turn TMR1 on
    mov_f1 INTCON,11000000b     ; Enable Global and Peripheral Interrupts

    return                     ; Return to caller

;-----
; Parse command from input from serial port
; From call to return, this function takes 215 cycles (worst case) or
; 43 usecs.
;-----


parse
    cjne_f1 arbit_flag,0,droutine
    inc arbit_flag             ; This indicates start of downloading

```

```

        ; and arbitration flag so background
        ; will not read FLASH
call    rword
mov_f1 flash_low,0
mov_f1 flash_high,8
mov_ff dsize_low,data_low
mov_ff dsize_high,data_high
cjbe_f1 data_high,15,nowrite
clr    dsize_high
clr    data_high
nowrite
call    writeFLASH
call    sendchar
return

droutine
call    rword
call    writeFLASH
call    sendchar
return

;-----
; Write word to FLASH memory
;-----

writeFLASH
        mov_wf w,flash_low           ; Move flash_low into w
        setb   STATUS,RP1            ; Switch to Memory Bank 2
        mov_fw EEADR,w              ; Move w into EEADR
        clrb   STATUS,RP1            ; Switch to Memory Bank 0

        mov_wf w,flash_high          ; Move flash_high into w
        setb   STATUS,RP1            ; Switch to Memory Bank 2
        mov_fw EEADRH,w             ; Move w into EEADR

        clrb   STATUS,RP1            ; Switch to Memory Bank 0
        mov_wf w,data_low            ; Move data_low into w
        setb   STATUS,RP1            ; Switch to Memory Bank 2
        mov_fw EEDATA,w              ; Move w into EEDATA

        clrb   STATUS,RP1            ; Switch to Memory Bank 0
        mov_wf w,data_high            ; Move data_high into w
        setb   STATUS,RP1            ; Switch to Memory Bank 2
        mov_fw EEDATH,w              ; Move w into EEDATA

        setb   STATUS,RP0              ; Switch to Memory Bank 3
        setb   EECON1,EEPGD           ; Point to program memory
        setb   EECON1,WREN2           ; Enable writes

        clrb   INTCON,GIE            ; Disable interrupts

        mov_wl w,0x55
        mov_fw EECON2,w
        mov_wl w,0xaa
        mov_fw EECON2,w
        setb   EECON1,WR              ; Start write operation

        nop
        nop                         ; 2 nops required

        clrb   EECON1,WREN2           ; Disable writes

        clrb   STATUS,RP0              ; Switch to Memory Bank 0
        clrb   STATUS,RP1

        inc    flash_low
        cjne_f1 flash_low,0,waround
        inc    flash_high             ; Increment flash_high when flash_low == 0

waround
        cjne_ff flash_high,dsize_high,wfreturn

```

```

    cjne_ff flash_low,dsize_low,wfreturn

    mov_fl flash_low,1           ; Seed for start of loop again
    mov_fl flash_high,8
    dec    arbit_flag          ; Data download complete...enable
    setb   INTCON,GIE         ; handler again
    setb   INTCON,GIE         ; Enable interrupts

wfreturn
    return

;-----
; rword receives a word or 2 bytes from serial port
;-----

rword
    mov_ff data_low,RCREG      ; Read byte from serial port
    mov_ff xmt_byte,data_low   ; Get ready to echo back data_low
    call   sendchar

waitchar
    sb     PIR1,RCIF
    jmp   waitchar
    mov_ff data_high,RCREG    ; Read byte from serial port
    mov_ff xmt_byte,data_high ; Get ready to echo back data_high

    return

;-----
; sendchar to serial port
;-----

sendchar
    sb     PIR1,TXIF          ; skip if PIR1,TXIF bit set
    jmp   sendchar

    mov_ff TXREG,xmt_byte     ; Place 'A' in transmit buffer

    return

;-----
; This routine reads a 14bit word from FLASH memory. The first call to
; this routine must set the FLASH start address found in 'flash_low'
; and 'flash_high'. This address is subsequently incremented at the
; end of this routine and is compared to a stop address found in
; 'dsize_low' and 'dsize_high'. This address must also be set before
; calling this routine.
;-----
readflash
    mov_wf w,flash_low         ; Move flash_low into w
    setb   STATUS,RP1           ; Switch to Memory Bank 2
    mov_fw EEADR,w              ; Move w into EEADR
    clrb   STATUS,RP1           ; Switch to Memory Bank 0

    mov_wf w,flash_high        ; Move flash_high into w
    setb   STATUS,RP1           ; Switch to Memory Bank 2
    mov_fw EEADRH,w             ; Move w into EEADR

    setb   STATUS,RP0           ; Switch to Memory Bank 3
    setb   EECON1,EEPGD         ; Point to Program Memory
    setb   EECON1,RD             ; Start read operation
    nop                            ; 2 nops required
    nop

    clrb   STATUS,RP0           ; Switch back to Memory Bank 2
    mov_wf w,EEDATA             ; Get low byte of FLASH data to w
    clrb   STATUS,RP1           ; Switch back to Memory Bank 0
    mov_fw data_low,w            ; Move w into data_low

    setb   STATUS,RP1           ; Switch back to Memory Bank 2
    mov_wf w,EEDATH             ; Get low byte of FLASH data to w

```

```

        clrb    STATUS,RP1           ; Switch back to Memory Bank 0
        mov fw  data_high,w         ; Move w into data_low

        inc     flash_low
        cjne fl  flash_low,0,around
        inc     flash_high          ; Increment flash_high when flash_low == 0

around
        cjne ff  flash_high,dsize_high,rfreturn
        cjne ff  flash_low,dsize_low,rfreturn
        mov fl   flash_low,1         ; Seed for start of loop again
        mov fl   flash_high,8

rfreturn
        return                  ; Return to calling function

;-----
; Beginning of designated, dynamic FLASH memory.
;-----
org    0x800
da    0x9f5           ; 100 5-tuples of data are below
                      ; This number is 100*5+2049

da    989             ; This is the default data which is
da    1445            ; from the file 181222fl_adas_aol.bin.mat
da    2168            ; These 100 points are subsamples from
da    2804            ; said file.
da    2831
da    1026
da    1439
da    2087
da    2793
da    2892
da    995
da    1450
da    2142
da    2825
da    2825
da    963
da    1466
da    2203
da    2838
da    2769
da    951
da    1468
da    2281
da    2926
da    2611
da    938
da    1520
da    2310
da    3018
da    2451
da    986
da    1702
da    2459
da    3166
da    1924
da    1058
da    1804
da    2555
da    3160
da    1661
da    1111
da    1871
da    2644
da    3108
da    1504
da    1086

```

da	1844
da	2651
da	3102
da	1556
da	1132
da	1880
da	2705
da	3065
da	1456
da	1132
da	1867
da	2700
da	3072
da	1467
da	1129
da	1911
da	2717
da	3045
da	1435
da	1155
da	1953
da	2731
da	3025
da	1373
da	1163
da	1941
da	2729
da	3032
da	1372
da	1174
da	1933
da	2707
da	3053
da	1370
da	1075
da	1832
da	2647
da	3102
da	1581
da	1052
da	1817
da	2637
da	3104
da	1627
da	1048
da	1808
da	2631
da	3107
da	1643
da	1030
da	1775
da	2616
da	3112
da	1704
da	1004
da	1735
da	2604
da	3105
da	1790
da	1006
da	1758
da	2615
da	3100
da	1759
da	1027
da	1817
da	2650
da	3084
da	1659
da	1030
da	1823

da	2656
da	3081
da	1648
da	1068
da	1873
da	2666
da	3079
da	1552
da	1036
da	1833
da	2654
da	3083
da	1631
da	1112
da	1930
da	2704
da	3047
da	1445
da	1107
da	1921
da	2703
da	3049
da	1457
da	1125
da	1952
da	2703
da	3044
da	1414
da	1188
da	2002
da	2707
da	3030
da	1311
da	1421
da	2233
da	2739
da	2856
da	987
da	1416
da	2217
da	2725
da	2882
da	998
da	1402
da	2181
da	2728
da	2904
da	1022
da	1350
da	2156
da	2708
da	2946
da	1077
da	1272
da	2044
da	2686
da	3029
da	1206
da	1243
da	2031
da	2683
da	3038
da	1242
da	1229
da	2028
da	2692
da	3033
da	1256
da	1243
da	2045
da	2696

da	3022
da	1231
da	1258
da	2081
da	2711
da	2992
da	1195
da	1323
da	2128
da	2720
da	2955
da	1112
da	1452
da	2284
da	2743
da	2810
da	949
da	1510
da	2337
da	2742
da	2754
da	895
da	1581
da	2382
da	2707
da	2727
da	840
da	1548
da	2379
da	2721
da	2729
da	861
da	1521
da	2354
da	2739
da	2739
da	883
da	1563
da	2377
da	2744
da	2701
da	853
da	1568
da	2368
da	2750
da	2701
da	852
da	1595
da	2410
da	2725
da	2680
da	828
da	1562
da	2384
da	2743
da	2697
da	852
da	1542
da	2375
da	2751
da	2703
da	866
da	1577
da	2380
da	2743
da	2693
da	843
da	1556
da	2377
da	2745
da	2702

da	857
da	1566
da	2381
da	2740
da	2701
da	850
da	1581
da	2392
da	2747
da	2679
da	839
da	1604
da	2423
da	2764
da	2624
da	823
da	1679
da	2462
da	2773
da	2542
da	781
da	1711
da	2484
da	2763
da	2516
da	764
da	1766
da	2527
da	2748
da	2458
da	738
da	1778
da	2532
da	2734
da	2463
da	731
da	1800
da	2523
da	2724
da	2470
da	721
da	1794
da	2514
da	2732
da	2473
da	724
da	1826
da	2522
da	2731
da	2447
da	712
da	1841
da	2533
da	2725
da	2433
da	707
da	1861
da	2541
da	2718
da	2419
da	699
da	1866
da	2534
da	2716
da	2425
da	697
da	1872
da	2512
da	2718
da	2441
da	694

da	1833
da	2498
da	2736
da	2460
da	710
da	1833
da	2508
da	2729
da	2459
da	709
da	1839
da	2518
da	2738
da	2434
da	709
da	1818
da	2511
da	2740
da	2453
da	716
da	1816
da	2510
da	2732
da	2464
da	716
da	1811
da	2508
da	2726
da	2476
da	717
da	1785
da	2498
da	2739
da	2488
da	729
da	1767
da	2487
da	2724
da	2525
da	734
da	1764
da	2487
da	2739
da	2511
da	737
da	1757
da	2488
da	2746
da	2506
da	741
da	1756
da	2493
da	2744
da	2502
da	741
da	1715
da	2459
da	2748
da	2554
da	761
da	1714
da	2453
da	2749
da	2560
da	762
da	1713
da	2453
da	2754
da	2555
da	763
da	1729

da	2451
da	2753
da	2549
da	755
da	1723
da	2454
da	2753
da	2550
da	758
da	1752
da	2459
da	2747
da	2535
da	744
da	1768
da	2474
da	2731
da	2530
da	735
da	1773
da	2473
da	2738
da	2520
da	733
da	1726
da	2459
da	2754
da	2542
da	757
da	1713
da	2452
da	2761
da	2548
da	764
da	1706
da	2451
da	2746
da	2569
da	765
da	1650
da	2414
da	2756
da	2620
da	798
da	1649
da	2412
da	2751
da	2628
da	798
da	1659
da	2416
da	2755
da	2614
da	793
da	1683
da	2426
da	2761
da	2588
da	780
da	1682
da	2434
da	2756
da	2586
da	779
da	1738
da	2472
da	2738
da	2541
da	749
da	1819
da	2542

da 2740
da 2418
da 718
da 2028
da 2632
da 2644
da 2277
da 657
da 2126
da 2667
da 2616
da 2179
da 650
da 2189
da 2664
da 2589
da 2155
da 641
da 2159
da 2640
da 2604
da 2197
da 638
da 2190
da 2654
da 2586
da 2170
da 637

end

Appendix C - Tables

The following table was produced by the Matlab code titled ‘digital3_100.m’. The first two columns (from left) of this table are cut and pasted into microcontroller code which runs on the custom DAC. The second column is a 12 bit number which the DAC translates to a voltage. This column is commutated in groups of 5 representing 5 channels of input. There are 500 rows of data which make up 100 data points because of the commutation. This second column is also the basis for the top plot of Figure 6. Digital counts are simply translated into voltage.

The third column represents the known chlorophyll concentration (commutated) in terms of voltage. The fourth column is the neural calculated chlorophyll concentration (commutated) also in terms of voltage. Both of these columns are shown in the bottom plot of Figure 6.

da	725	0.235843	0.210782
da	1295	0.235843	0.210782
da	2198	0.235843	0.210782
da	2993	0.235843	0.210782
da	3027	0.235843	0.210782
da	771	0.201260	0.219559
da	1287	0.201260	0.219559
da	2097	0.201260	0.219559
da	2980	0.201260	0.219559
da	3103	0.201260	0.219559
da	732	0.215398	0.211185
da	1301	0.215398	0.211185
da	2165	0.215398	0.211185
da	3020	0.215398	0.211185
da	3020	0.215398	0.211185
da	692	0.236492	0.204417
da	1320	0.236492	0.204417
da	2241	0.236492	0.204417
da	3035	0.236492	0.204417
da	2949	0.236492	0.204417
da	677	0.167827	0.173936
da	1323	0.167827	0.173936
da	2339	0.167827	0.173936
da	3146	0.167827	0.173936
da	2752	0.167827	0.173936
da	661	0.132224	0.153172
da	1388	0.132224	0.153172
da	2376	0.132224	0.153172
da	3261	0.132224	0.153172
da	2551	0.132224	0.153172
da	720	0.091181	0.078783
da	1616	0.091181	0.078783
da	2562	0.091181	0.078783
da	3446	0.091181	0.078783
da	1893	0.091181	0.078783
da	810	0.073359	0.048088
da	1743	0.073359	0.048088
da	2682	0.073359	0.048088
da	3438	0.073359	0.048088
da	1564	0.073359	0.048088
da	877	0.042238	0.035614
da	1826	0.042238	0.035614
da	2793	0.042238	0.035614
da	3373	0.042238	0.035614
da	1368	0.042238	0.035614
da	845	0.030411	0.039255

da	1793	0.030411	0.039255
da	2801	0.030411	0.039255
da	3365	0.030411	0.039255
da	1433	0.030411	0.039255
da	903	0.027668	0.032895
da	1838	0.027668	0.032895
da	2869	0.027668	0.032895
da	3319	0.027668	0.032895
da	1309	0.027668	0.032895
da	903	0.028887	0.032857
da	1822	0.028887	0.032857
da	2863	0.028887	0.032857
da	3328	0.028887	0.032857
da	1322	0.028887	0.032857
da	899	0.021711	0.033262
da	1877	0.021711	0.033262
da	2885	0.021711	0.033262
da	3295	0.021711	0.033262
da	1281	0.021711	0.033262
da	932	0.026557	0.030773
da	1930	0.026557	0.030773
da	2902	0.026557	0.030773
da	3269	0.026557	0.030773
da	1204	0.026557	0.030773
da	942	0.022445	0.029988
da	1915	0.022445	0.029988
da	2900	0.022445	0.029988
da	3278	0.022445	0.029988
da	1203	0.022445	0.029988
da	955	0.027250	0.028740
da	1905	0.027250	0.028740
da	2872	0.027250	0.028740
da	3305	0.027250	0.028740
da	1200	0.027250	0.028740
da	832	0.032196	0.041074
da	1779	0.032196	0.041074
da	2797	0.032196	0.041074
da	3366	0.032196	0.041074
da	1464	0.032196	0.041074
da	804	0.034486	0.045331
da	1760	0.034486	0.045331
da	2785	0.034486	0.045331
da	3368	0.034486	0.045331
da	1522	0.034486	0.045331
da	798	0.033874	0.046417
da	1749	0.033874	0.046417
da	2777	0.033874	0.046417
da	3372	0.033874	0.046417
da	1542	0.033874	0.046417
da	776	0.038268	0.051015
da	1707	0.038268	0.051015
da	2758	0.038268	0.051015
da	3378	0.038268	0.051015
da	1619	0.038268	0.051015
da	743	0.049951	0.058450
da	1656	0.049951	0.058450
da	2743	0.049951	0.058450
da	3370	0.049951	0.058450
da	1725	0.049951	0.058450
da	745	0.047763	0.056946
da	1685	0.047763	0.056946
da	2756	0.047763	0.056946
da	3364	0.047763	0.056946
da	1687	0.047763	0.056946
da	772	0.038793	0.049894
da	1759	0.038793	0.049894
da	2800	0.038793	0.049894
da	3344	0.038793	0.049894
da	1562	0.038793	0.049894
da	775	0.035659	0.049134
da	1767	0.035659	0.049134

da	2808	0.035659	0.049134
da	3339	0.035659	0.049134
da	1549	0.035659	0.049134
da	823	0.036185	0.041881
da	1830	0.036185	0.041881
da	2821	0.036185	0.041881
da	3337	0.036185	0.041881
da	1428	0.036185	0.041881
da	783	0.038727	0.047911
da	1780	0.038727	0.047911
da	2806	0.038727	0.047911
da	3342	0.038727	0.047911
da	1527	0.038727	0.047911
da	878	0.029958	0.035275
da	1901	0.029958	0.035275
da	2868	0.029958	0.035275
da	3297	0.029958	0.035275
da	1294	0.029958	0.035275
da	872	0.030263	0.035805
da	1889	0.030263	0.035805
da	2867	0.030263	0.035805
da	3300	0.030263	0.035805
da	1309	0.030263	0.035805
da	895	0.028727	0.033721
da	1928	0.028727	0.033721
da	2867	0.028727	0.033721
da	3293	0.028727	0.033721
da	1255	0.028727	0.033721
da	973	0.025518	0.027632
da	1990	0.025518	0.027632
da	2871	0.025518	0.027632
da	3276	0.025518	0.027632
da	1127	0.025518	0.027632
da	1265	0.017845	0.017343
da	2280	0.017845	0.017343
da	2912	0.017845	0.017343
da	3059	0.017845	0.017343
da	722	0.017845	0.017343
da	1258	0.017565	0.016825
da	2260	0.017565	0.016825
da	2894	0.017565	0.016825
da	3090	0.017565	0.016825
da	736	0.017565	0.016825
da	1241	0.026697	0.017389
da	2215	0.026697	0.017389
da	2899	0.026697	0.017389
da	3118	0.026697	0.017389
da	765	0.026697	0.017389
da	1176	0.018451	0.018295
da	2183	0.018451	0.018295
da	2874	0.018451	0.018295
da	3170	0.018451	0.018295
da	834	0.018451	0.018295
da	1078	0.036322	0.021297
da	2043	0.036322	0.021297
da	2846	0.036322	0.021297
da	3275	0.036322	0.021297
da	995	0.036322	0.021297
da	1041	0.029221	0.023055
da	2027	0.029221	0.023055
da	2842	0.029221	0.023055
da	3286	0.029221	0.023055
da	1041	0.029221	0.023055
da	1025	0.027160	0.024163
da	2023	0.027160	0.024163
da	2853	0.027160	0.024163
da	3279	0.027160	0.024163
da	1058	0.027160	0.024163
da	1042	0.028849	0.023308
da	2044	0.028849	0.023308
da	2858	0.028849	0.023308

da	3265	0.028849	0.023308
da	1027	0.028849	0.023308
da	1061	0.024842	0.022854
da	2090	0.024842	0.022854
da	2877	0.024842	0.022854
da	3228	0.024842	0.022854
da	982	0.024842	0.022854
da	1142	0.023006	0.019796
da	2148	0.023006	0.019796
da	2888	0.023006	0.019796
da	3182	0.023006	0.019796
da	878	0.023006	0.019796
da	1303	0.018828	0.016850
da	2343	0.018828	0.016850
da	2917	0.018828	0.016850
da	3001	0.018828	0.016850
da	674	0.018828	0.016850
da	1375	0.019838	0.015802
da	2409	0.019838	0.015802
da	2916	0.019838	0.015802
da	2930	0.019838	0.015802
da	607	0.019838	0.015802
da	1465	0.014758	0.013125
da	2465	0.014758	0.013125
da	2872	0.014758	0.013125
da	2897	0.014758	0.013125
da	539	0.014758	0.013125
da	1423	0.015635	0.014271
da	2461	0.015635	0.014271
da	2889	0.015635	0.014271
da	2899	0.015635	0.014271
da	565	0.015635	0.014271
da	1390	0.016203	0.015514
da	2431	0.016203	0.015514
da	2912	0.016203	0.015514
da	2912	0.016203	0.015514
da	592	0.016203	0.015514
da	1442	0.016916	0.015222
da	2459	0.016916	0.015222
da	2918	0.016916	0.015222
da	2865	0.016916	0.015222
da	554	0.016916	0.015222
da	1448	0.018790	0.015472
da	2448	0.018790	0.015472
da	2925	0.018790	0.015472
da	2864	0.018790	0.015472
da	553	0.018790	0.015472
da	1482	0.016482	0.013997
da	2501	0.016482	0.013997
da	2894	0.016482	0.013997
da	2838	0.016482	0.013997
da	523	0.016482	0.013997
da	1440	0.016382	0.015228
da	2468	0.016382	0.015228
da	2917	0.016382	0.015228
da	2859	0.016382	0.015228
da	553	0.016382	0.015228
da	1416	0.014566	0.015864
da	2457	0.014566	0.015864
da	2927	0.014566	0.015864
da	2867	0.014566	0.015864
da	571	0.014566	0.015864
da	1460	0.015208	0.015056
da	2463	0.015208	0.015056
da	2917	0.015208	0.015056
da	2855	0.015208	0.015056
da	542	0.015208	0.015056
da	1434	0.015915	0.015390
da	2460	0.015915	0.015390
da	2920	0.015915	0.015390
da	2866	0.015915	0.015390

da	559	0.015915	0.015390
da	1445	0.014994	0.015000
da	2464	0.014994	0.015000
da	2913	0.014994	0.015000
da	2864	0.014994	0.015000
da	551	0.014994	0.015000
da	1465	0.015795	0.015218
da	2478	0.015795	0.015218
da	2921	0.015795	0.015218
da	2837	0.015795	0.015218
da	537	0.015795	0.015218
da	1493	0.014588	0.016081
da	2517	0.014588	0.016081
da	2943	0.014588	0.016081
da	2768	0.014588	0.016081
da	517	0.014588	0.016081
da	1587	0.017821	0.016413
da	2566	0.017821	0.016413
da	2954	0.017821	0.016413
da	2666	0.017821	0.016413
da	464	0.017821	0.016413
da	1627	0.016700	0.015877
da	2593	0.016700	0.015877
da	2942	0.016700	0.015877
da	2633	0.016700	0.015877
da	443	0.016700	0.015877
da	1696	0.017182	0.015254
da	2647	0.017182	0.015254
da	2923	0.017182	0.015254
da	2561	0.017182	0.015254
da	411	0.017182	0.015254
da	1711	0.015443	0.014436
da	2653	0.015443	0.014436
da	2905	0.015443	0.014436
da	2567	0.015443	0.014436
da	402	0.015443	0.014436
da	1738	0.015304	0.013859
da	2642	0.015304	0.013859
da	2893	0.015304	0.013859
da	2576	0.015304	0.013859
da	389	0.015304	0.013859
da	1731	0.017281	0.014323
da	2631	0.017281	0.014323
da	2903	0.017281	0.014323
da	2579	0.017281	0.014323
da	393	0.017281	0.014323
da	1770	0.016189	0.014455
da	2640	0.016189	0.014455
da	2902	0.016189	0.014455
da	2546	0.016189	0.014455
da	379	0.016189	0.014455
da	1789	0.015902	0.014196
da	2654	0.015902	0.014196
da	2894	0.015902	0.014196
da	2529	0.015902	0.014196
da	372	0.015902	0.014196
da	1815	0.016210	0.013934
da	2664	0.016210	0.013934
da	2885	0.016210	0.013934
da	2512	0.016210	0.013934
da	362	0.016210	0.013934
da	1820	0.014746	0.013852
da	2656	0.014746	0.013852
da	2883	0.014746	0.013852
da	2519	0.014746	0.013852
da	359	0.014746	0.013852
da	1828	0.014775	0.013971
da	2628	0.014775	0.013971
da	2886	0.014775	0.013971
da	2540	0.014775	0.013971
da	356	0.014775	0.013971

da	1779	0.014543	0.014788
da	2611	0.014543	0.014788
da	2909	0.014543	0.014788
da	2563	0.014543	0.014788
da	376	0.014543	0.014788
da	1779	0.011627	0.014344
da	2623	0.011627	0.014344
da	2900	0.011627	0.014344
da	2561	0.011627	0.014344
da	374	0.011627	0.014344
da	1787	0.011406	0.014963
da	2636	0.011406	0.014963
da	2910	0.011406	0.014963
da	2531	0.011406	0.014963
da	374	0.011406	0.014963
da	1760	0.011025	0.014934
da	2626	0.011025	0.014934
da	2913	0.011025	0.014934
da	2554	0.011025	0.014934
da	384	0.011025	0.014934
da	1758	0.011341	0.014398
da	2626	0.011341	0.014398
da	2903	0.011341	0.014398
da	2568	0.011341	0.014398
da	383	0.011341	0.014398
da	1752	0.013332	0.013995
da	2624	0.013332	0.013995
da	2895	0.013332	0.013995
da	2583	0.013332	0.013995
da	384	0.013332	0.013995
da	1719	0.012274	0.014637
da	2610	0.012274	0.014637
da	2912	0.012274	0.014637
da	2598	0.012274	0.014637
da	399	0.012274	0.014637
da	1697	0.011312	0.013620
da	2597	0.011312	0.013620
da	2893	0.011312	0.013620
da	2645	0.011312	0.013620
da	406	0.011312	0.013620
da	1693	0.012107	0.014506
da	2596	0.012107	0.014506
da	2911	0.012107	0.014506
da	2627	0.012107	0.014506
da	410	0.012107	0.014506
da	1684	0.015358	0.014929
da	2598	0.015358	0.014929
da	2921	0.015358	0.014929
da	2621	0.015358	0.014929
da	415	0.015358	0.014929
da	1684	0.013587	0.014841
da	2605	0.013587	0.014841
da	2918	0.013587	0.014841
da	2616	0.013587	0.014841
da	414	0.013587	0.014841
da	1632	0.013317	0.014921
da	2562	0.013317	0.014921
da	2924	0.013317	0.014921
da	2680	0.013317	0.014921
da	439	0.013317	0.014921
da	1630	0.014232	0.014953
da	2554	0.014232	0.014953
da	2924	0.014232	0.014953
da	2688	0.014232	0.014953
da	441	0.014232	0.014953
da	1630	0.013365	0.015224
da	2554	0.013365	0.015224
da	2930	0.013365	0.015224
da	2682	0.013365	0.015224
da	441	0.013365	0.015224
da	1650	0.012826	0.015216

da	2552	0.012826	0.015216
da	2929	0.012826	0.015216
da	2674	0.012826	0.015216
da	432	0.012826	0.015216
da	1641	0.013743	0.015208
da	2556	0.013743	0.015208
da	2929	0.013743	0.015208
da	2675	0.013743	0.015208
da	436	0.013743	0.015208
da	1678	0.012462	0.014963
da	2562	0.012462	0.014963
da	2922	0.012462	0.014963
da	2657	0.012462	0.014963
da	418	0.012462	0.014963
da	1698	0.013579	0.014026
da	2581	0.013579	0.014026
da	2902	0.013579	0.014026
da	2650	0.013579	0.014026
da	407	0.013579	0.014026
da	1704	0.013984	0.014468
da	2579	0.013984	0.014468
da	2910	0.013984	0.014468
da	2639	0.013984	0.014468
da	405	0.013984	0.014468
da	1645	0.013598	0.015295
da	2562	0.013598	0.015295
da	2931	0.013598	0.015295
da	2666	0.013598	0.015295
da	434	0.013598	0.015295
da	1629	0.015899	0.015693
da	2553	0.015899	0.015693
da	2940	0.015899	0.015693
da	2673	0.015899	0.015693
da	443	0.015899	0.015693
da	1621	0.013093	0.014748
da	2552	0.013093	0.014748
da	2920	0.013093	0.014748
da	2700	0.013093	0.014748
da	445	0.013093	0.014748
da	1550	0.013068	0.015343
da	2506	0.013068	0.015343
da	2933	0.013068	0.015343
da	2763	0.013068	0.015343
da	486	0.013068	0.015343
da	1550	0.012666	0.015066
da	2503	0.012666	0.015066
da	2927	0.012666	0.015066
da	2773	0.012666	0.015066
da	486	0.012666	0.015066
da	1562	0.013861	0.015311
da	2508	0.013861	0.015311
da	2932	0.013861	0.015311
da	2756	0.013861	0.015311
da	479	0.013861	0.015311
da	1592	0.013871	0.015604
da	2521	0.013871	0.015604
da	2939	0.013871	0.015604
da	2723	0.013871	0.015604
da	463	0.013871	0.015604
da	1590	0.014613	0.015313
da	2531	0.014613	0.015313
da	2933	0.014613	0.015313
da	2721	0.014613	0.015313
da	462	0.014613	0.015313
da	1660	0.013553	0.014339
da	2578	0.013553	0.014339
da	2910	0.013553	0.014339
da	2665	0.013553	0.014339
da	424	0.013553	0.014339
da	1762	0.012476	0.015093
da	2666	0.012476	0.015093

da	2914	0.012476	0.015093
da	2511	0.012476	0.015093
da	385	0.012476	0.015093
da	2024	0.012960	0.011630
da	2778	0.012960	0.011630
da	2793	0.012960	0.011630
da	2334	0.012960	0.011630
da	309	0.012960	0.011630
da	2145	0.011863	0.011771
da	2821	0.011863	0.011771
da	2759	0.011863	0.011771
da	2212	0.011863	0.011771
da	301	0.011863	0.011771
da	2225	0.012235	0.011147
da	2818	0.012235	0.011147
da	2724	0.012235	0.011147
da	2181	0.012235	0.011147
da	289	0.012235	0.011147
da	2186	0.011849	0.011314
da	2788	0.011849	0.011314
da	2744	0.011849	0.011314
da	2234	0.011849	0.011314
da	285	0.011849	0.011314
da	2226	0.012543	0.010845
da	2806	0.012543	0.010845
da	2720	0.012543	0.010845
da	2201	0.012543	0.010845
da	284	0.012543	0.010845

The following table comes from the file titled ‘p1n914_measure.txt’. This table was derived by measuring the input (left column) and output (right column) voltages of our hardware transfer function isolated in a test circuit. The ‘Net’ portion of Figure 4 is said circuit. A calibrated voltage source was stepped through the voltage range of –10 to 10 volts. The top plot of Figure 5 is a comparison of this data (measured d1n914) with both a simulated d1n914 arrangement and the tansig transfer function created in Matlab. As shown, all three functions are virtually identical in the linear region of the curve. Differences occur at various regions above and below around +- 0.5 volts input.

Accurately modeling the break point from linear to non-linear and the functions non-linear shape was the criteria for measuring response of this circuit. This data is used in both neural training and validation as a proxy for the Matlab tansig transfer function. Both Matlab functions ‘diodesig3.m and deltadiode3.m’ were created to use this data.

```

-1.0000000e+001 -7.1700000e-001
-9.0000000e+000 -7.1100000e-001
-8.0000000e+000 -7.0400000e-001
-7.0000000e+000 -6.9700000e-001
-6.0000000e+000 -6.8800000e-001
-5.0000000e+000 -6.7700000e-001
-4.0000000e+000 -6.6400000e-001
-3.0000000e+000 -6.4700000e-001
-2.0000000e+000 -6.2200000e-001
-1.5000000e+000 -6.0200000e-001
-1.4000000e+000 -5.9600000e-001
-1.3000000e+000 -5.9100000e-001
-1.2000000e+000 -5.8400000e-001
-1.1000000e+000 -5.7700000e-001
-1.0000000e+000 -5.6800000e-001
-8.0000000e-001 -5.4300000e-001
-6.0000000e-001 -4.9800000e-001
-4.0000000e-001 -3.8600000e-001
-2.0000000e-001 -1.9800000e-001
0.0000000e+000 0.0000000e+000
2.0000000e-001 1.9800000e-001
4.0000000e-001 3.8600000e-001
6.0000000e-001 4.9800000e-001
8.0000000e-001 5.4300000e-001
1.0000000e+000 5.6800000e-001
1.1000000e+000 5.7700000e-001
1.2000000e+000 5.8400000e-001
1.3000000e+000 5.9100000e-001
1.4000000e+000 5.9600000e-001
1.5000000e+000 6.0200000e-001
2.0000000e+000 6.2200000e-001
3.0000000e+000 6.4700000e-001
4.0000000e+000 6.6400000e-001
5.0000000e+000 6.7700000e-001
6.0000000e+000 6.8800000e-001
7.0000000e+000 6.9700000e-001
8.0000000e+000 7.0400000e-001
9.0000000e+000 7.1100000e-001
1.0000000e+001 7.1700000e-001

```

The following table comes from the file titled ‘d1n914_circuit3.txt’. This table was derived by measuring the input (left column) and output (right column) voltages of our hardware transfer function while residing in the complete circuit of Figure 8. A calibrated voltage source was stepped through the voltage range of -9 to 9 volts applied equally to all 6 inputs (includes bias). This input voltage range was chosen because it represented the limits which could be generated by the complementary DAC board

Like the table described above, accurately modeling the break point from linear to non-linear and the functions non-linear shape was the criteria for measuring response of this circuit. This data is used in both neural training and validation as a proxy for the Matlab tansig transfer function. Both Matlab functions ‘diodesig6.m’ and ‘deltadiode6.m’ were created to use this data.

```

-9.000000e+000 -5.800000e-001
-8.000000e+000 -5.730000e-001
-7.000000e+000 -5.660000e-001
-6.000000e+000 -5.570000e-001
-5.000000e+000 -5.470000e-001
-4.000000e+000 -5.350000e-001
-3.000000e+000 -5.190000e-001
-2.000000e+000 -4.950000e-001
-1.500000e+000 -4.760000e-001
-1.400000e+000 -4.710000e-001
-1.300000e+000 -4.680000e-001
-1.200000e+000 -4.620000e-001
-1.100000e+000 -4.550000e-001
-1.000000e+000 -4.480000e-001
-8.000000e-001 -4.290000e-001
-6.000000e-001 -3.990000e-001
-4.000000e-001 -3.380000e-001
-2.000000e-001 -1.910000e-001
0.000000e+000 0.000000e+000
2.000000e-001 1.910000e-001
4.000000e-001 3.380000e-001
6.000000e-001 3.990000e-001
8.000000e-001 4.290000e-001
1.000000e+000 4.480000e-001
1.100000e+000 4.550000e-001
1.200000e+000 4.620000e-001
1.300000e+000 4.680000e-001
1.400000e+000 4.710000e-001
1.500000e+000 4.760000e-001
2.000000e+000 4.950000e-001
3.000000e+000 5.190000e-001
4.000000e+000 5.350000e-001
5.000000e+000 5.470000e-001
6.000000e+000 5.570000e-001
7.000000e+000 5.660000e-001
8.000000e+000 5.730000e-001
9.000000e+000 5.800000e-001

```

The following table comes from the file titled ‘003853ad.cal_new’. This table was generated by the Airborne Oceanographic Lidar (AOL) group at Wallops Flight Facility. The contents are a calibration file for ADAS. The following is a description of each column. Starting from the left, the first column is simply an index from 0-255. The second column is wavelength (in nm) ranging from 398.0-737.3 nm with centers 1.3 nm apart. The third column is a scaling factor that when divided into radiance, returns the original (uncalibrated), 12bit counts recorded from ADAS’s analog-to-digital converter. This step is key because actual spectrometer output (if coupled to an AANN) would not be calibrated. Columns 4-6 are unused.

0	398.0	0.00008814	1823.83	3.7062	0.16076
1	399.3	0.00008993	1827.04	3.8580	0.16430
2	400.7	0.00009233	1818.38	3.7489	0.16790
3	402.0	0.00052001	329.89	3.7329	0.17155
4	403.3	0.00054997	318.65	3.7487	0.17524
5	404.6	0.00056889	314.62	3.8845	0.17898
6	406.0	0.00059355	307.92	3.8168	0.18277
7	407.3	0.00061731	302.28	3.6920	0.18660
8	408.6	0.00064038	297.44	3.6084	0.19047
9	410.0	0.00065422	297.13	3.5423	0.19439
10	411.3	0.00065629	302.27	3.6085	0.19838
11	412.6	0.00064844	312.15	3.6891	0.20241
12	414.0	0.00063002	327.74	3.6495	0.20649
13	415.3	0.00060947	345.56	3.6572	0.21061
14	416.6	0.00058588	366.59	3.6497	0.21478
15	418.0	0.00056501	387.58	3.6379	0.21899
16	419.3	0.00055103	405.14	3.6553	0.22324
17	420.6	0.00054606	416.65	3.6221	0.22751
18	421.9	0.00055057	421.00	3.5861	0.23179
19	423.3	0.00056260	419.68	3.4904	0.23611
20	424.6	0.00058281	412.60	3.5814	0.24047
21	425.9	0.00060496	404.77	3.6022	0.24487
22	427.3	0.00062777	397.13	3.6611	0.24931
23	428.6	0.00064619	392.75	3.6337	0.25379
24	429.9	0.00065829	392.39	3.5461	0.25831
25	431.2	0.00066203	397.11	3.5452	0.26290
26	432.6	0.00065762	406.81	3.5629	0.26753
27	433.9	0.00064625	421.20	3.5239	0.27220
28	435.2	0.00063054	439.17	3.4458	0.27691
29	436.6	0.00061155	460.57	3.5951	0.28166
30	437.9	0.00059492	481.49	3.5846	0.28645
31	439.2	0.00058201	500.47	3.5942	0.29127
32	440.6	0.00057153	518.12	3.4856	0.29612
33	441.9	0.00056655	531.25	3.4823	0.30098
34	443.2	0.00056515	541.23	3.5252	0.30588
35	444.5	0.00056743	547.75	3.4159	0.31081
36	445.9	0.00057218	551.88	3.5021	0.31577
37	447.2	0.00057963	553.41	3.3805	0.32077
38	448.5	0.00058714	554.91	3.3720	0.32581
39	449.9	0.00059371	557.30	3.4486	0.33087
40	451.2	0.00059984	560.17	3.5991	0.33602
41	452.5	0.00060217	566.61	3.6244	0.34120
42	453.9	0.00060057	576.80	3.5305	0.34641
43	455.2	0.00059525	590.77	3.4313	0.35165
44	456.5	0.00058721	607.84	3.5310	0.35693
45	457.9	0.00057739	627.38	3.4618	0.36224
46	459.2	0.00056847	646.62	3.5787	0.36758
47	460.5	0.00056155	664.13	3.5512	0.37294
48	461.8	0.00055718	678.94	3.5724	0.37829
49	463.2	0.00055384	692.75	3.4908	0.38367
50	464.5	0.00055335	703.15	3.6631	0.38909
51	465.8	0.00055188	714.88	3.4218	0.39452
52	467.2	0.00055077	726.24	3.6563	0.39999
53	468.5	0.00054880	738.86	3.5081	0.40548
54	469.8	0.00054510	754.00	3.4619	0.41100
55	471.1	0.00054113	769.78	3.3570	0.41655

56	472.5	0.00053673	786.46	3.4614	0.42212
57	473.8	0.00053520	799.17	3.5638	0.42771
58	475.1	0.00053382	811.76	3.5531	0.43333
59	476.5	0.00053145	826.00	3.5426	0.43898
60	477.8	0.00052802	842.09	3.4648	0.44465
61	479.1	0.00052254	861.81	3.3694	0.45034
62	480.5	0.00051998	877.09	3.4587	0.45607
63	481.8	0.00051743	892.61	3.4846	0.46187
64	483.1	0.00051554	907.16	3.4814	0.46768
65	484.5	0.00051536	918.82	3.5740	0.47353
66	485.8	0.00051795	925.54	3.5220	0.47939
67	487.1	0.00052314	927.61	3.5097	0.48527
68	488.4	0.00052989	926.95	3.5090	0.49118
69	489.8	0.00053643	926.68	3.5881	0.49710
70	491.1	0.00054141	929.04	3.5015	0.50299
71	492.4	0.00054332	936.62	3.4660	0.50889
72	493.8	0.00054219	949.48	3.3964	0.51480
73	495.1	0.00053718	969.38	3.5022	0.52073
74	496.4	0.00053001	993.71	3.4376	0.52668
75	497.8	0.00052238	1019.64	3.5187	0.53264
76	499.1	0.00051443	1047.02	3.5796	0.53861
77	500.4	0.00050696	1074.29	3.5413	0.54463
78	501.7	0.00049996	1101.49	3.6147	0.55070
79	503.1	0.00049405	1127.00	3.4542	0.55680
80	504.4	0.00048831	1152.77	3.5364	0.56290
81	505.7	0.00048421	1175.17	3.6189	0.56903
82	507.1	0.00048184	1193.68	3.5031	0.57516
83	508.4	0.00048192	1206.22	3.5985	0.58131
84	509.7	0.00048426	1213.13	3.4685	0.58747
85	511.0	0.00048864	1214.62	3.4957	0.59351
86	512.4	0.00049564	1209.62	3.4566	0.59954
87	513.7	0.00050487	1199.45	3.4692	0.60557
88	515.0	0.00051574	1185.86	3.5546	0.61160
89	516.4	0.00052804	1169.70	3.5010	0.61765
90	517.7	0.00054051	1153.89	3.6008	0.62370
91	519.0	0.00055198	1140.89	3.6667	0.62975
92	520.4	0.00056262	1130.13	3.5027	0.63583
93	521.7	0.00057079	1124.74	3.5721	0.64198
94	523.0	0.00057621	1124.83	3.4498	0.64814
95	524.3	0.00057871	1130.63	3.5716	0.65430
96	525.7	0.00057854	1141.61	3.5458	0.66046
97	527.0	0.00057307	1163.27	3.5639	0.66663
98	528.3	0.00056619	1188.31	3.5929	0.67281
99	529.7	0.00055662	1219.83	3.5928	0.67898
100	531.0	0.00054511	1257.04	3.6631	0.68523
101	532.3	0.00053269	1298.14	3.5546	0.69150
102	533.7	0.00051994	1342.05	3.6800	0.69778
103	535.0	0.00050797	1386.02	3.5892	0.70406
104	536.3	0.00049735	1428.26	3.6276	0.71034
105	537.7	0.00048877	1466.16	3.6630	0.71662
106	539.0	0.00048309	1496.41	3.7623	0.72291
107	540.3	0.00047988	1519.49	3.5942	0.72917
108	541.6	0.00047990	1532.31	3.6866	0.73536
109	543.0	0.00048243	1537.10	3.6474	0.74154
110	544.3	0.00048753	1533.70	3.5596	0.74773
111	545.6	0.00049463	1524.18	3.6524	0.75391
112	547.0	0.00050328	1510.27	3.5066	0.76009
113	548.3	0.00051222	1495.96	3.6810	0.76626
114	549.6	0.00052231	1478.88	3.5489	0.77243
115	551.0	0.00053364	1459.18	3.6105	0.77867
116	552.3	0.00054593	1437.80	3.6083	0.78493
117	553.6	0.00055886	1415.74	3.7093	0.79120
118	554.9	0.00057131	1395.83	3.6512	0.79745
119	556.3	0.00058157	1381.97	3.6887	0.80371
120	557.6	0.00058937	1374.28	3.6239	0.80995
121	558.9	0.00059377	1374.61	3.6682	0.81620
122	560.3	0.00059505	1382.08	3.6171	0.82241
123	561.6	0.00059374	1395.45	3.6336	0.82853
124	562.9	0.00058957	1415.68	3.6561	0.83464
125	564.2	0.00058325	1441.48	3.5759	0.84075
126	565.6	0.00057496	1472.86	3.6778	0.84684

127	566.9	0.00056552	1508.21	3.6431	0.85292
128	568.2	0.00055512	1547.41	3.7392	0.85900
129	569.6	0.00054481	1587.83	3.9056	0.86506
130	570.9	0.00053489	1628.60	3.6947	0.87111
131	572.2	0.00052592	1667.84	3.9505	0.87715
132	573.6	0.00051851	1703.32	3.6920	0.88318
133	574.9	0.00051276	1734.16	3.7815	0.88920
134	576.2	0.00050894	1758.95	3.7941	0.89521
135	577.5	0.00050776	1774.85	3.8274	0.90120
136	578.9	0.00050866	1783.47	3.9091	0.90718
137	580.2	0.00051095	1787.11	3.7859	0.91313
138	581.5	0.00051446	1786.26	3.6513	0.91896
139	582.9	0.00051907	1781.60	3.6456	0.92477
140	584.2	0.00052497	1772.61	3.7972	0.93057
141	585.5	0.00053038	1765.43	3.7526	0.93635
142	586.9	0.00053588	1758.07	3.7666	0.94212
143	588.2	0.00054133	1751.00	3.7322	0.94787
144	589.5	0.00054600	1746.52	3.6630	0.95360
145	590.8	0.00055083	1741.87	3.6850	0.95948
146	592.2	0.00055669	1734.22	3.8916	0.96543
147	593.5	0.00056499	1719.27	3.7235	0.97137
148	594.8	0.00057550	1698.16	3.6764	0.97729
149	596.2	0.00058609	1677.57	3.7726	0.98320
150	597.5	0.00059547	1661.02	3.7530	0.98909
151	598.8	0.00060209	1652.51	3.7798	0.99496
152	600.2	0.00060712	1648.42	3.7684	1.00079
153	601.5	0.00060878	1653.08	3.8047	1.00636
154	602.8	0.00060853	1662.88	3.7522	1.01191
155	604.2	0.00060630	1678.11	3.9344	1.01744
156	605.5	0.00060277	1697.09	3.7739	1.02295
157	606.8	0.00059855	1718.21	3.9215	1.02844
158	608.1	0.00059368	1741.52	3.6956	1.03391
159	609.5	0.00058892	1764.84	3.8501	1.03935
160	610.8	0.00058540	1784.86	3.9193	1.04486
161	612.1	0.00058403	1798.55	3.8792	1.05040
162	613.5	0.00058471	1805.89	4.0454	1.05592
163	614.8	0.00058345	1819.23	3.8951	1.06142
164	616.1	0.00057930	1841.70	4.0009	1.06690
165	617.5	0.00057388	1868.61	3.8417	1.07236
166	618.8	0.00056988	1891.26	3.9470	1.07779
167	620.1	0.00057012	1899.98	3.8972	1.08321
168	621.4	0.00057097	1906.85	3.9738	1.08875
169	622.8	0.00057202	1912.99	3.8963	1.09426
170	624.1	0.00057378	1916.69	3.7066	1.09975
171	625.4	0.00057633	1917.70	3.8357	1.10522
172	626.8	0.00057938	1917.01	3.8791	1.11067
173	628.1	0.00058315	1913.90	3.7826	1.11610
174	629.4	0.00058795	1907.46	3.8826	1.12150
175	630.8	0.00059346	1898.82	3.8731	1.12688
176	632.1	0.00059922	1889.51	4.0546	1.13224
177	633.4	0.00060474	1881.10	3.8399	1.13757
178	634.7	0.00060949	1875.15	3.9869	1.14288
179	636.1	0.00061274	1873.82	3.7756	1.14817
180	637.4	0.00061534	1874.47	3.8106	1.15344
181	638.7	0.00061752	1876.33	3.7502	1.15868
182	640.1	0.00061974	1878.01	3.8883	1.16388
183	641.4	0.00062333	1875.04	4.0342	1.16876
184	642.7	0.00062862	1866.98	3.8891	1.17362
185	644.0	0.00063524	1855.13	3.9182	1.17845
186	645.4	0.00064137	1844.86	3.8264	1.18325
187	646.7	0.00064710	1835.93	3.8774	1.18802
188	648.0	0.00065156	1830.64	3.9595	1.19277
189	649.4	0.00065439	1829.92	3.7414	1.19748
190	650.7	0.00065694	1830.21	3.7829	1.20234
191	652.0	0.00065855	1833.32	3.8940	1.20733
192	653.4	0.00066010	1836.53	3.8139	1.21229
193	654.7	0.00066121	1840.89	3.9470	1.21722
194	656.0	0.00066320	1842.77	3.9814	1.22213
195	657.3	0.00066450	1846.51	3.8889	1.22701
196	658.7	0.00066659	1848.02	3.9120	1.23186
197	660.0	0.00066901	1848.53	3.9427	1.23669

198	661.3	0.00067158	1848.37	3.9782	1.24133
199	662.7	0.00067387	1848.94	3.9502	1.24594
200	664.0	0.00067616	1849.44	3.7806	1.25052
201	665.3	0.00067799	1851.16	3.8973	1.25507
202	666.7	0.00067991	1852.59	3.8879	1.25960
203	668.0	0.00068128	1855.49	3.8845	1.26410
204	669.3	0.00068223	1859.45	3.9238	1.26857
205	670.7	0.00068278	1864.34	3.7735	1.27293
206	672.0	0.00068235	1871.73	3.8023	1.27717
207	673.3	0.00068229	1878.07	3.8746	1.28139
208	674.6	0.00068219	1884.48	3.8621	1.28557
209	676.0	0.00068120	1893.33	3.9056	1.28973
210	677.3	0.00068063	1900.94	3.8783	1.29385
211	678.6	0.00067975	1909.44	3.9009	1.29794
212	680.0	0.00067961	1915.82	3.7312	1.30201
213	681.3	0.00067948	1922.87	3.7900	1.30655
214	682.6	0.00068029	1927.24	3.9691	1.31108
215	684.0	0.00068126	1931.11	3.8682	1.31558
216	685.3	0.00068242	1934.36	3.9403	1.32006
217	686.6	0.00068431	1935.52	3.9756	1.32450
218	687.9	0.00068695	1934.54	3.9238	1.32893
219	689.3	0.00068918	1934.66	3.9225	1.33332
220	690.6	0.00069250	1931.33	3.7694	1.33745
221	691.9	0.00069495	1930.01	3.7559	1.34126
222	693.3	0.00069826	1926.26	3.8612	1.34503
223	694.6	0.00070176	1922.00	3.8305	1.34877
224	695.9	0.00070550	1917.08	3.6856	1.35249
225	697.2	0.00070932	1911.95	3.6351	1.35617
226	698.6	0.00071351	1905.81	3.7702	1.35982
227	699.9	0.00071801	1898.94	3.7427	1.36345
228	701.2	0.00072343	1890.36	3.9147	1.36755
229	702.6	0.00072886	1881.91	3.8732	1.37166
230	703.9	0.00073503	1871.68	3.8614	1.37574
231	705.2	0.00074161	1860.53	3.8022	1.37979
232	706.6	0.00074873	1848.23	3.7911	1.38382
233	707.9	0.00075604	1835.65	3.8381	1.38782
234	709.2	0.00076373	1822.37	3.7403	1.39179
235	710.6	0.00077237	1806.79	3.6713	1.39550
236	711.9	0.00078019	1792.97	3.7984	1.39886
237	713.2	0.00078876	1777.71	3.7248	1.40219
238	714.5	0.00079723	1762.97	3.7577	1.40549
239	715.9	0.00080558	1748.74	3.8516	1.40875
240	717.2	0.00081428	1734.04	3.6219	1.41199
241	718.5	0.00082246	1720.68	3.7228	1.41519
242	719.9	0.00083057	1707.70	3.8312	1.41837
243	721.2	0.00083916	1694.18	3.6400	1.42168
244	722.5	0.00084777	1680.87	3.8377	1.42499
245	723.8	0.00085670	1667.16	3.7011	1.42826
246	725.2	0.00086553	1653.90	3.6998	1.43151
247	726.5	0.00087345	1642.59	3.6597	1.43472
248	727.8	0.00087967	1634.60	3.7499	1.43791
249	729.2	0.00088139	1635.00	3.8386	1.44106
250	730.5	0.00087797	1645.17	3.5854	1.44440
251	731.8	0.00086980	1664.84	3.7101	1.44808
252	733.2	0.00085704	1693.88	3.6552	1.45172
253	734.5	0.00084107	1730.34	3.6652	1.45534
254	735.8	0.00082363	1771.36	3.6980	1.45894
255	737.2	0.00080768	1810.73	3.7581	1.46250

Appendix D - Coefficients

The following coefficients represent the neural network weights and biases of the active AANN.

W1 =

-0.1583	0.3721	0.2646	-0.3840	-0.0173
0.7156	-0.6733	0.9665	0.0227	-0.4232

B1 =

-1.1293
0.3359

W2 =

-0.2752	-0.4124
---------	---------

B2 =

0.1187

The following coefficients represent the neural network weights and biases of the passive AANN.

W1 =

0.3313	0.1775	0.4412	0.2359	0.2852
0.0052	0.9913	0	0.3837	0.8759

B1 =

-0.0010
-1.0712

W2 =

1	1
---	---

B2 =

0.4387

Appendix E - Photos

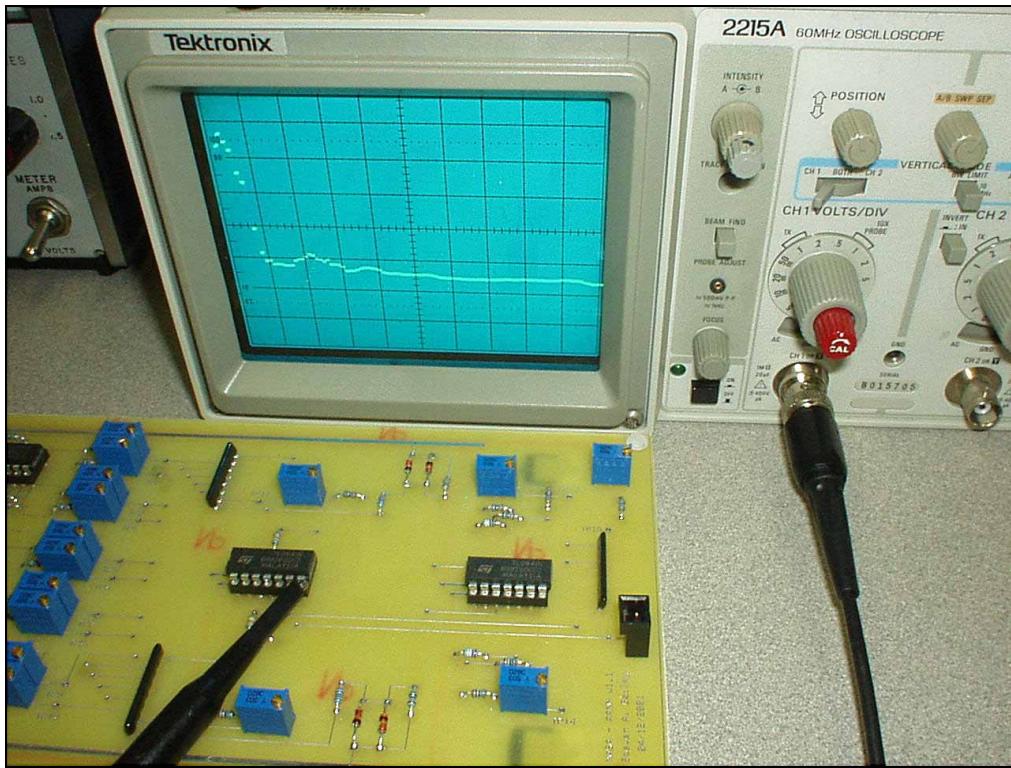


Figure 11 - Active AANN prototype showing scope output

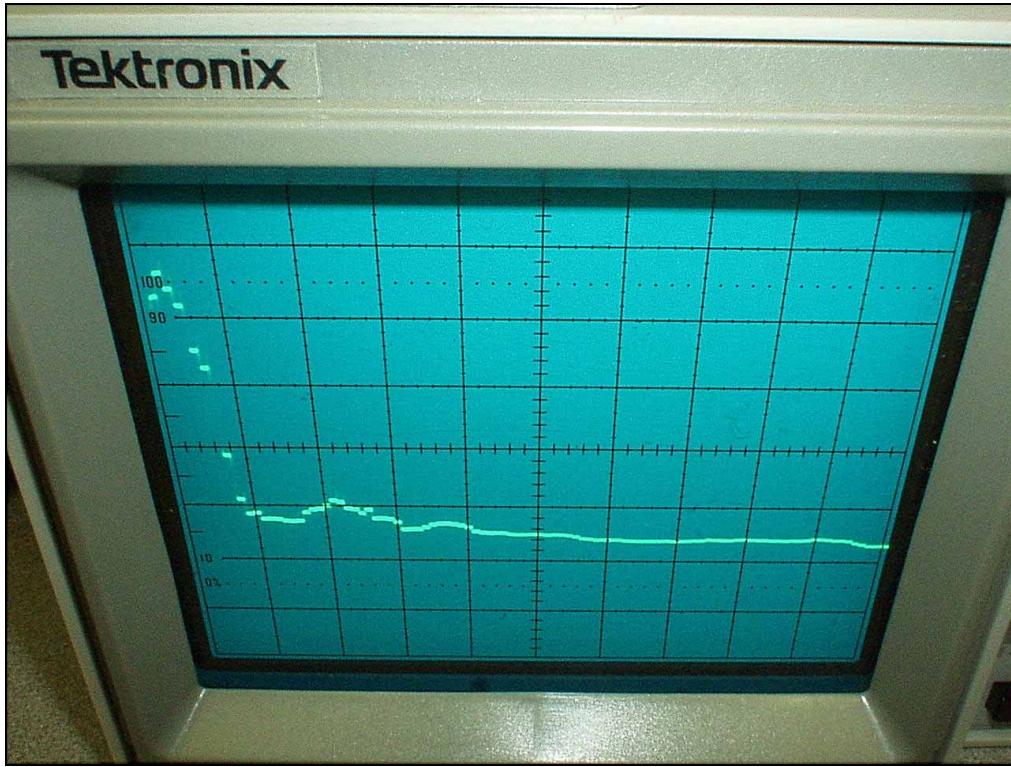


Figure 12 - Closeup of active AANN scope output

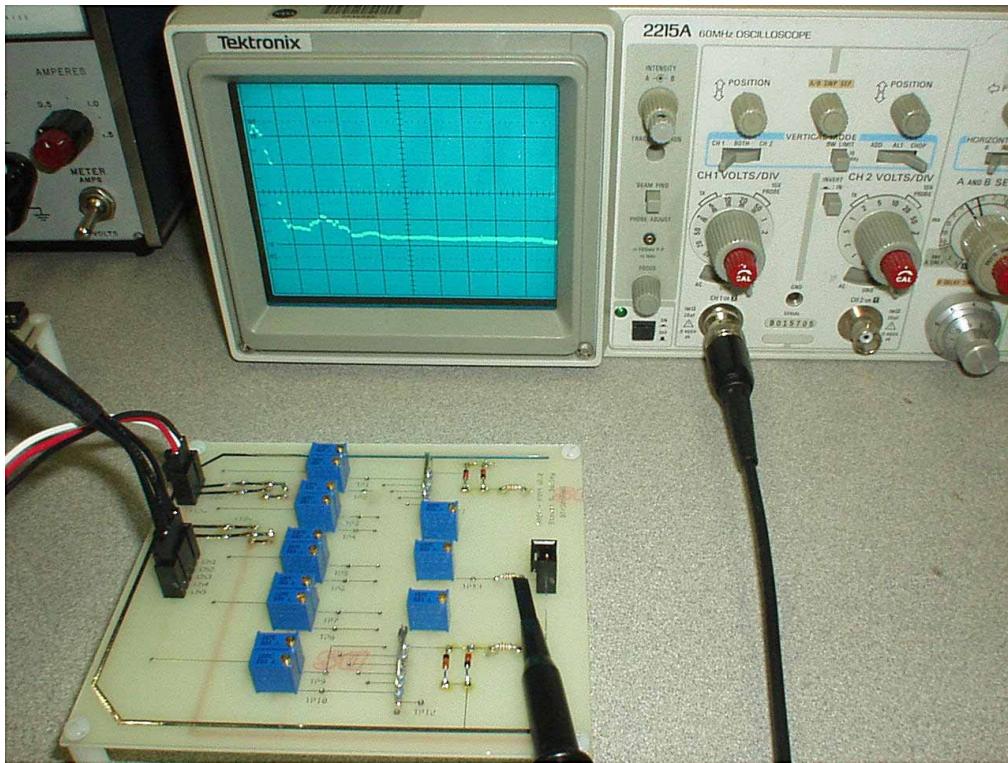


Figure 13 - Passive AANN prototype showing scope output

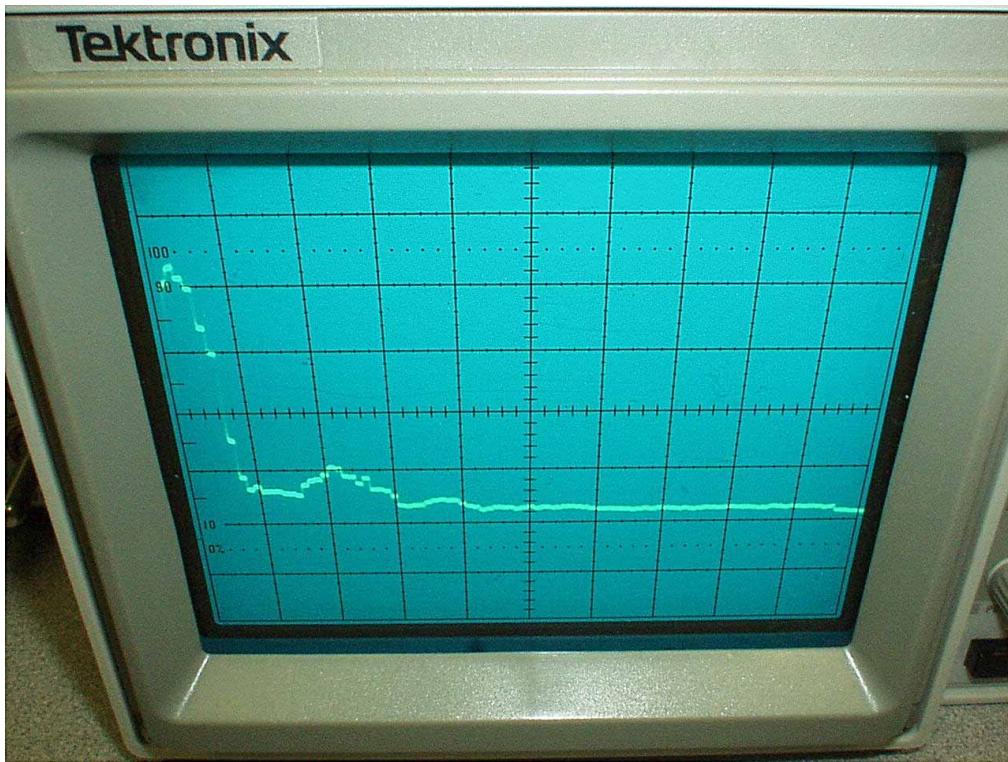


Figure 14 - Closeup of passive AANN scope output

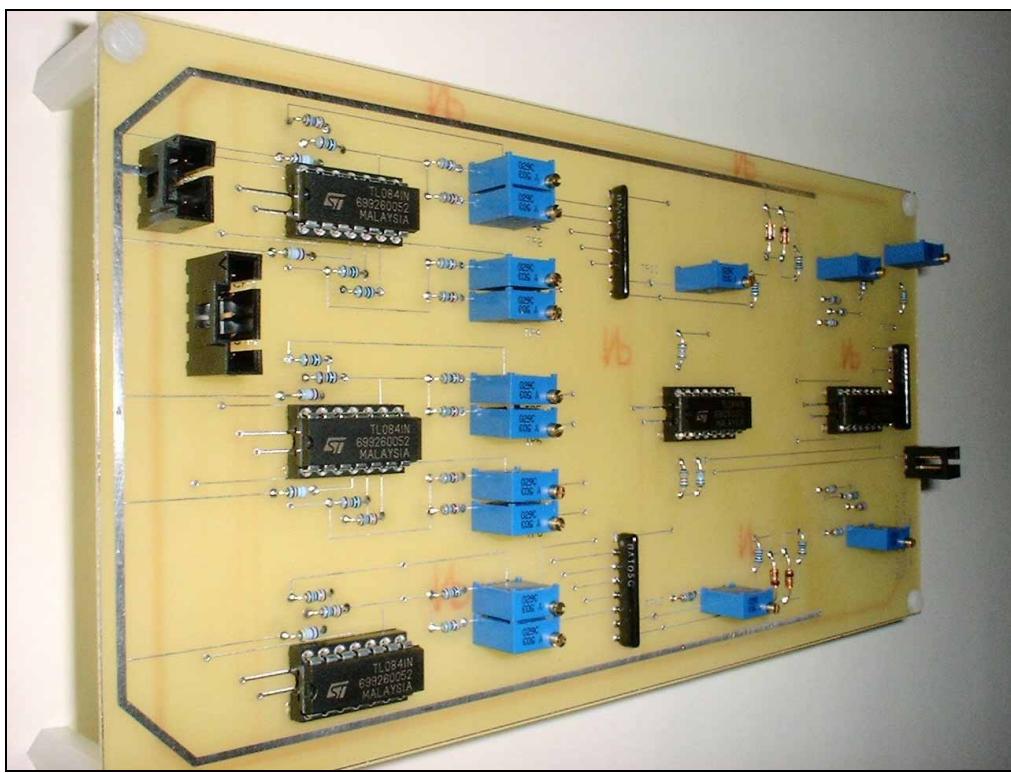


Figure 15 - Active AANN Prototype Board

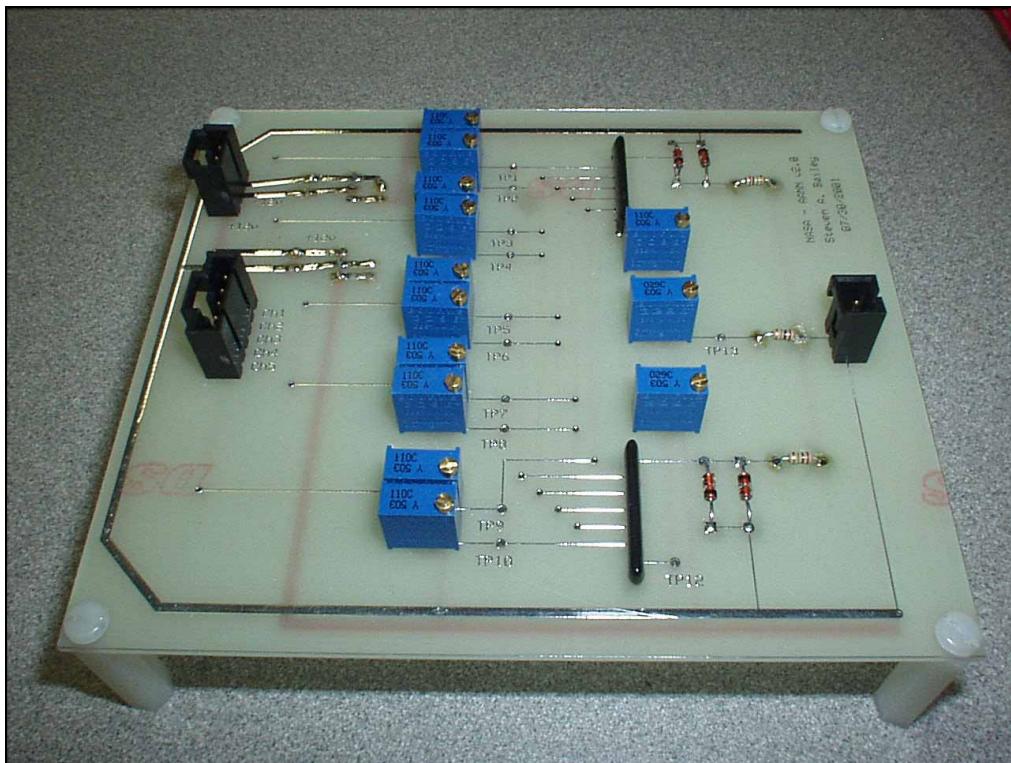


Figure 16 - Passive AANN Prototype Board

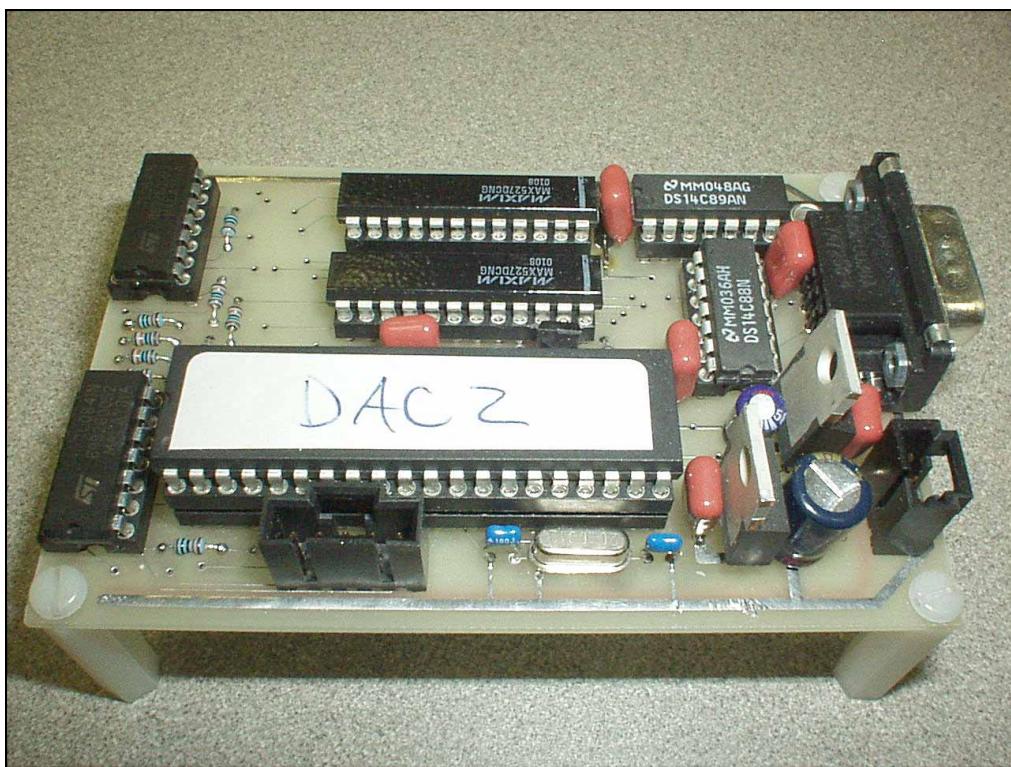


Figure 17 - Custom Digital-to-Analog (DAC) board

Appendix F - Schematics

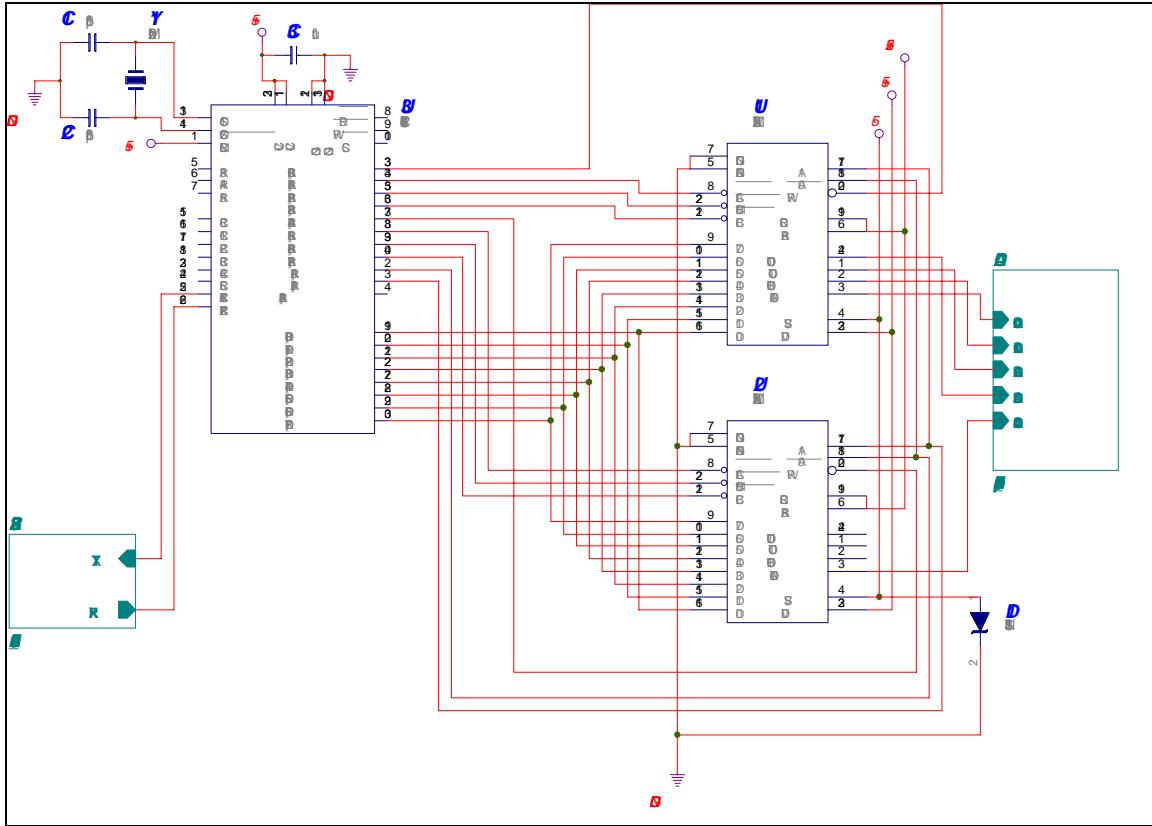


Figure 18 - DAC Schematic (Page 1)

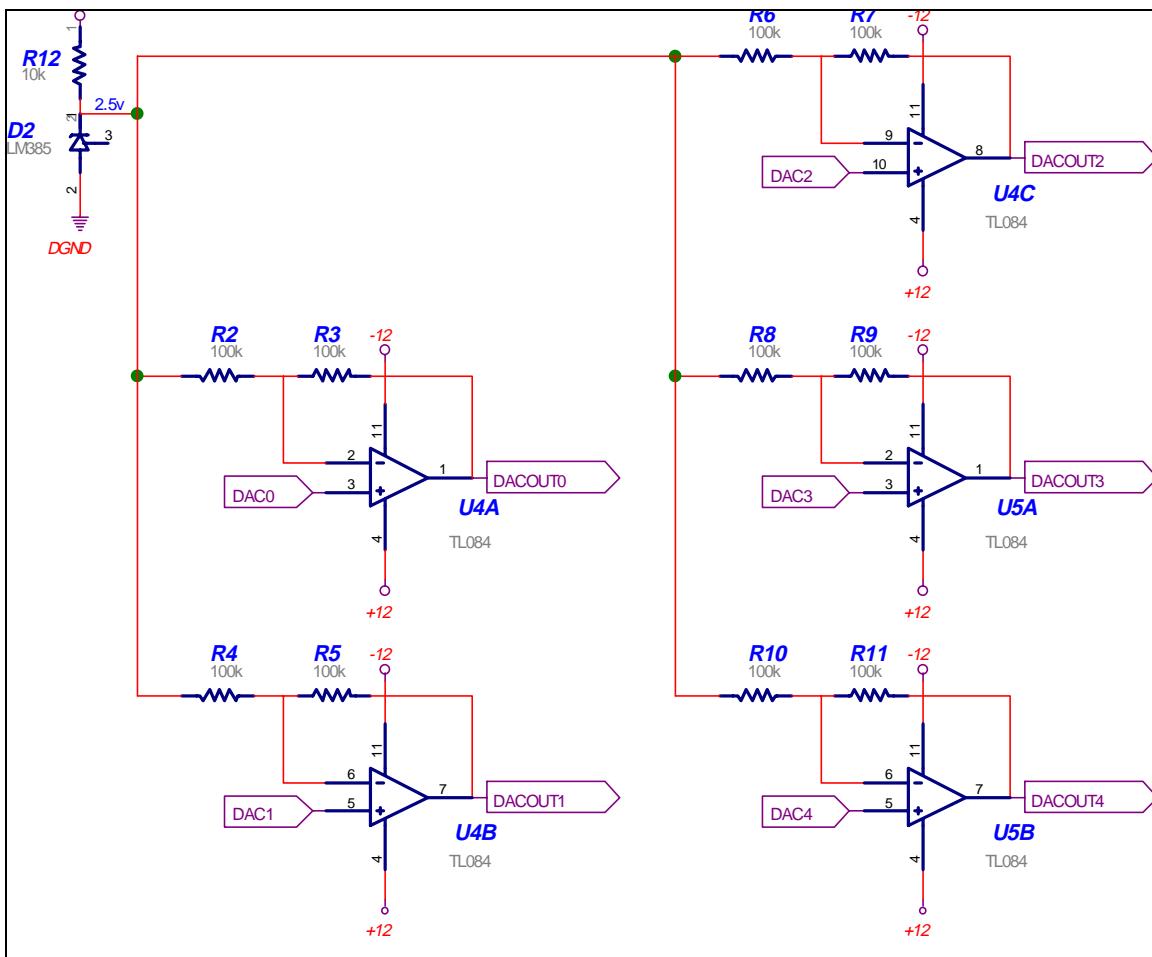


Figure 19 - DAC Schematic (Page 2)

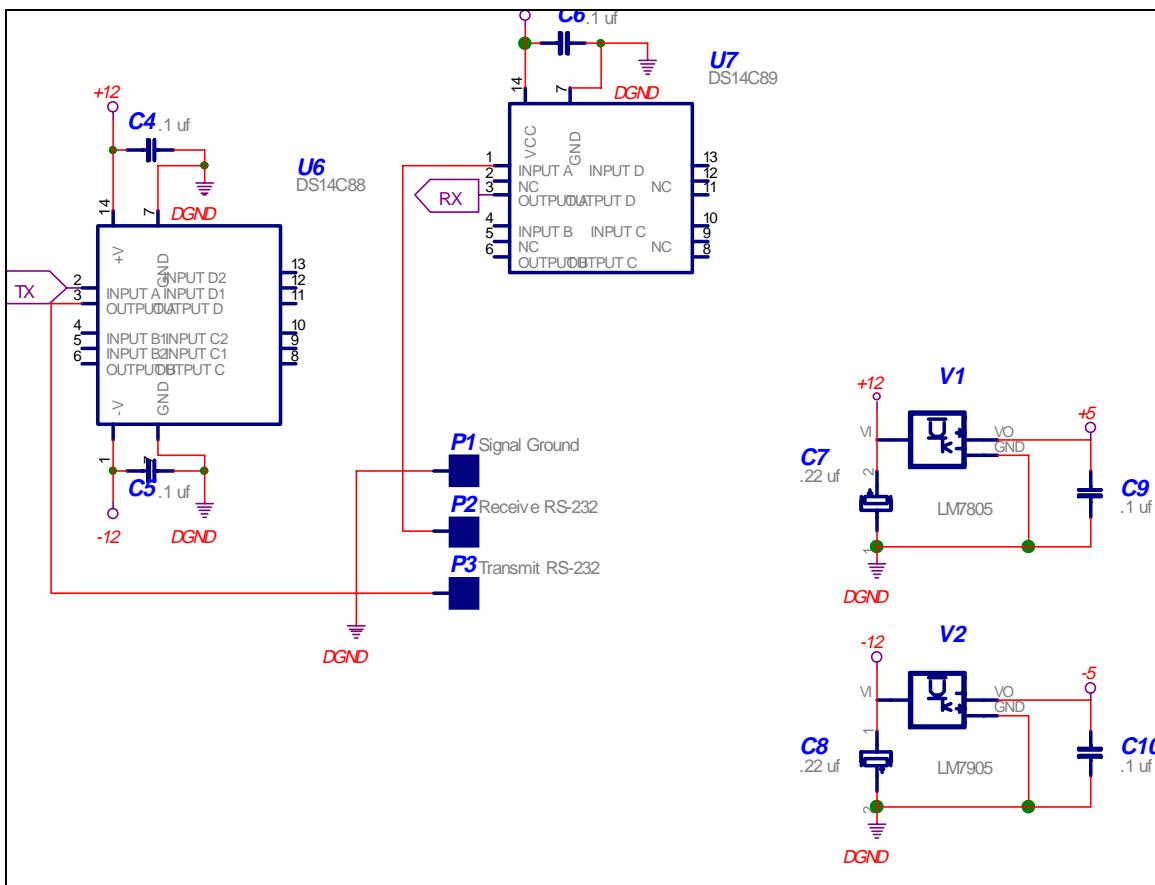


Figure 20 - DAC Schematic (Page 3)

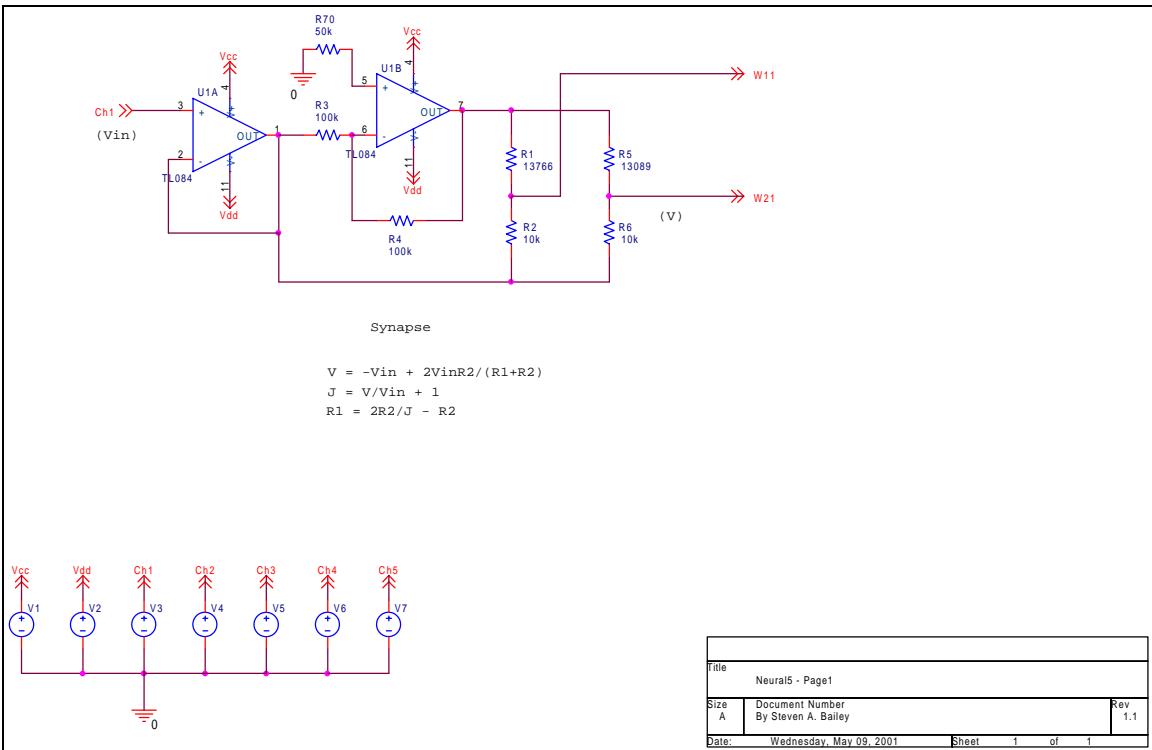


Figure 21 - Active AANN Schematic (Page 1)

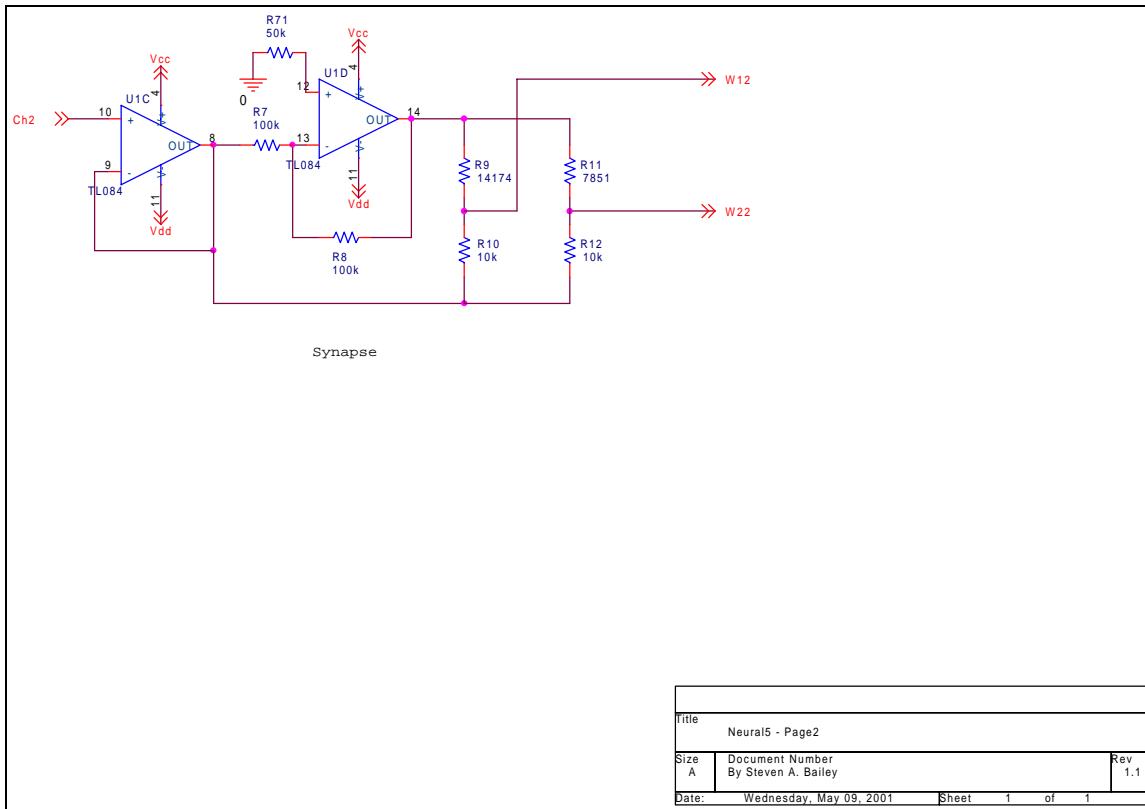


Figure 22 - Active AANN Schematic (Page 2)

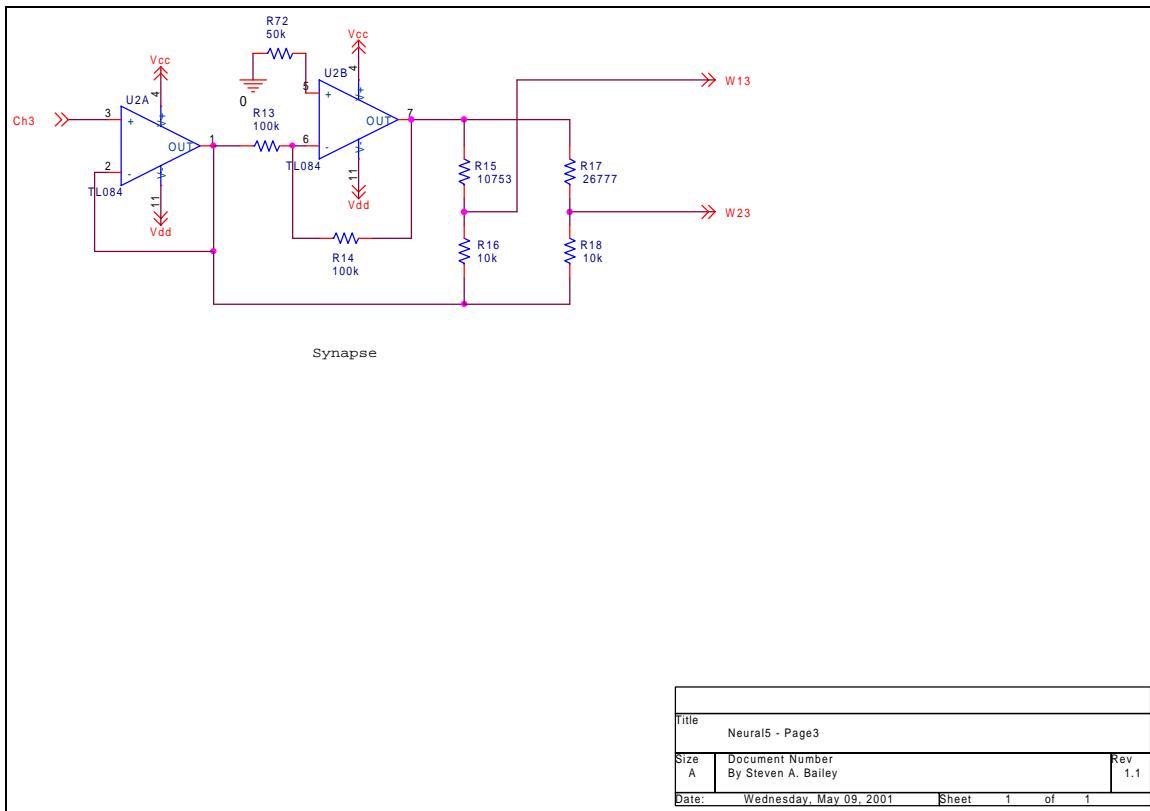


Figure 23 - Active AANN Schematic (Page 3)

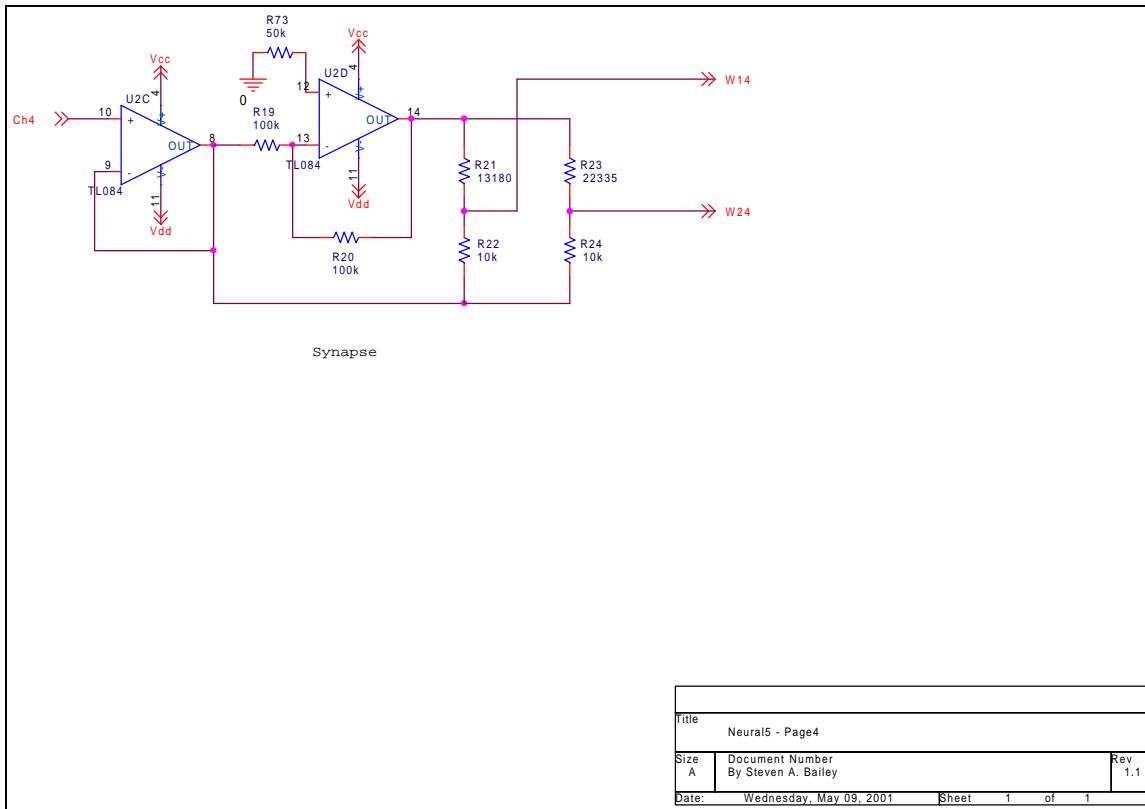


Figure 24 - Active AANN Schematic (Page 4)

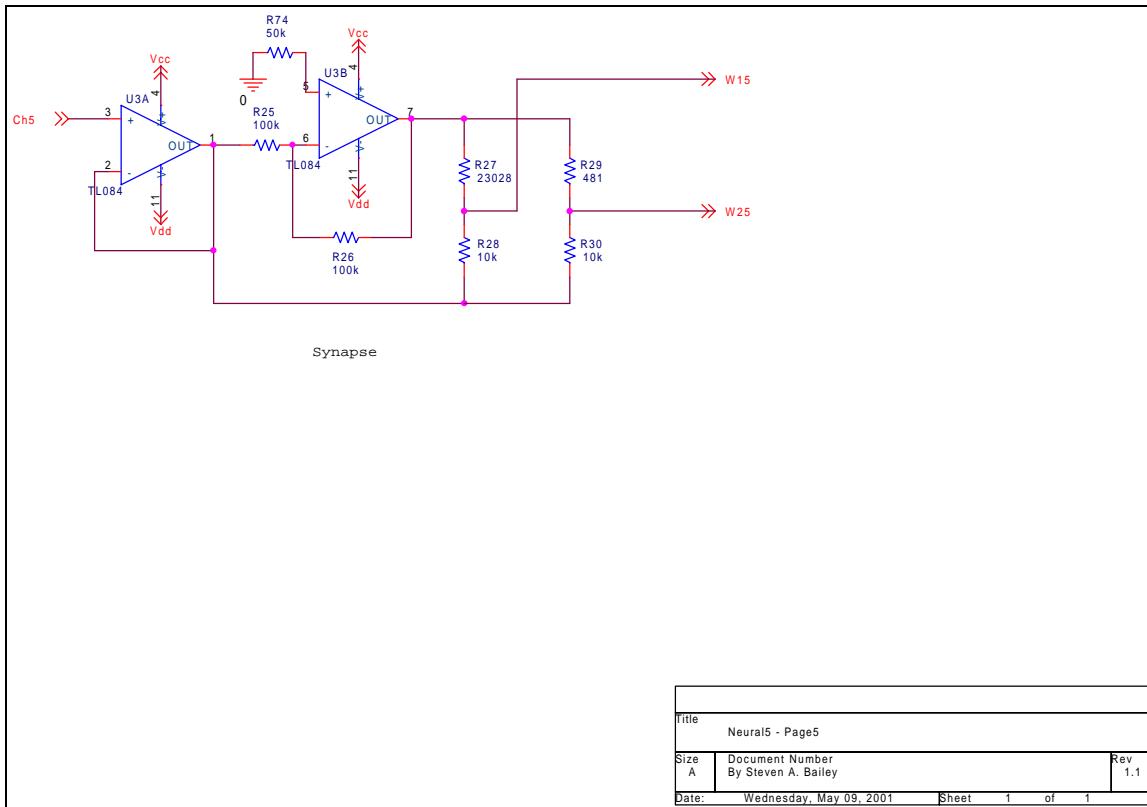


Figure 25 - Active AANN Schematic (Page 5)

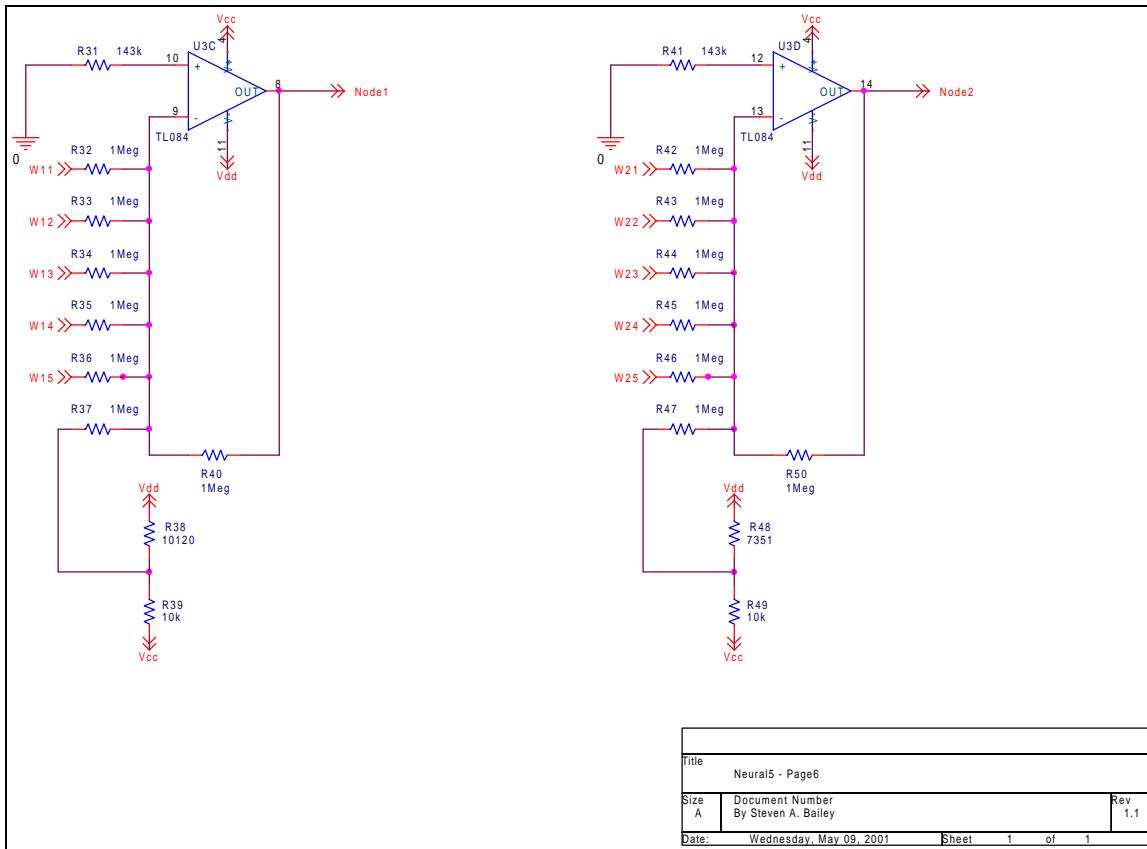
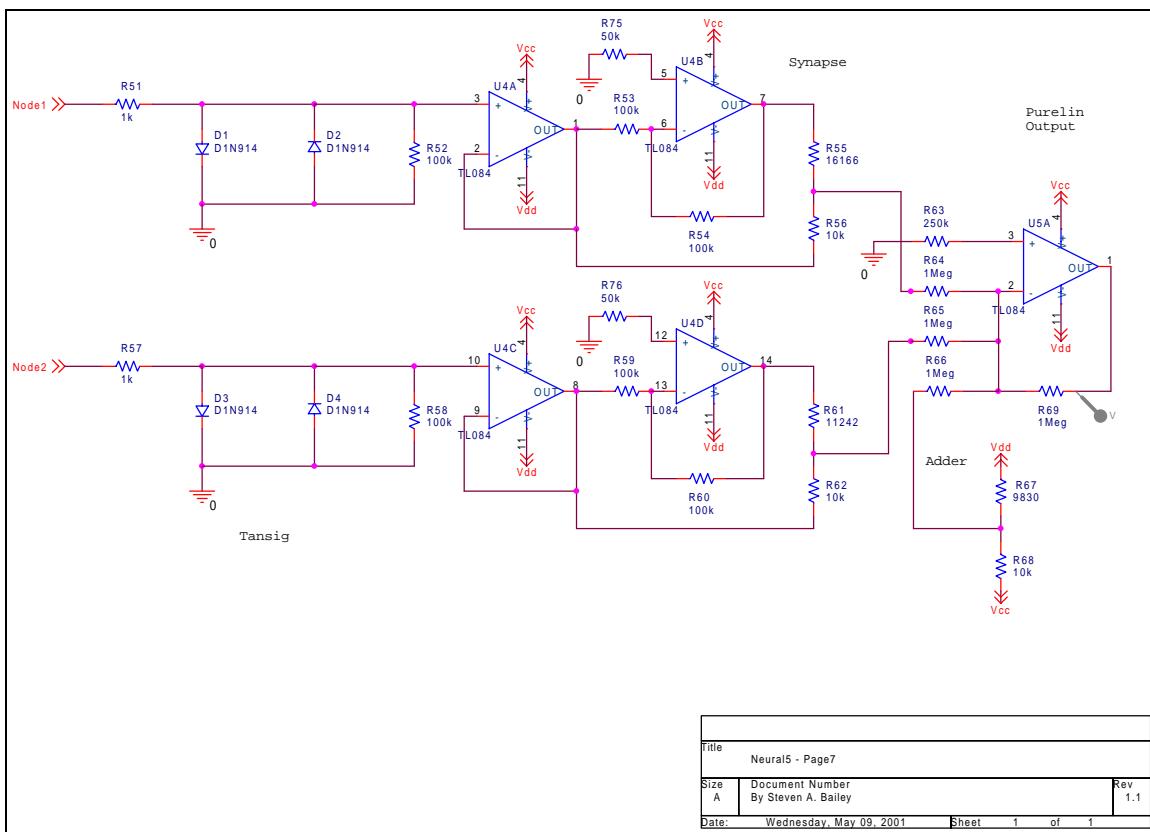


Figure 26 - Active AANN Schematic (Page 6)



Title		Neural5 - Page7	
Size	Document Number	Rev	
A	By Steven A. Bailey	1.1	

Date: Wednesday, May 09, 2001 Sheet 1 of 1

Figure 27 - Active AANN Schematic (Page 7)

Appendix G - Printed Circuit Boards (PCBs)

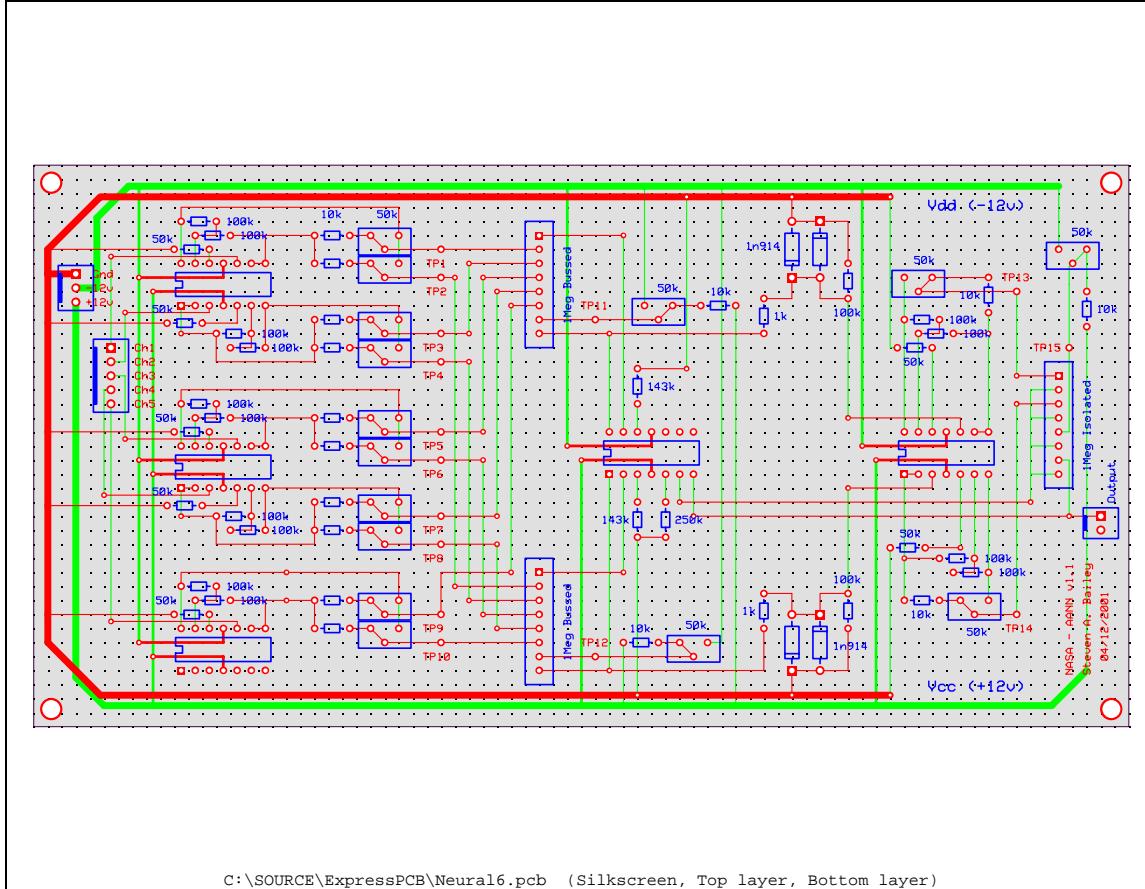


Figure 28 - Active AANN PCB

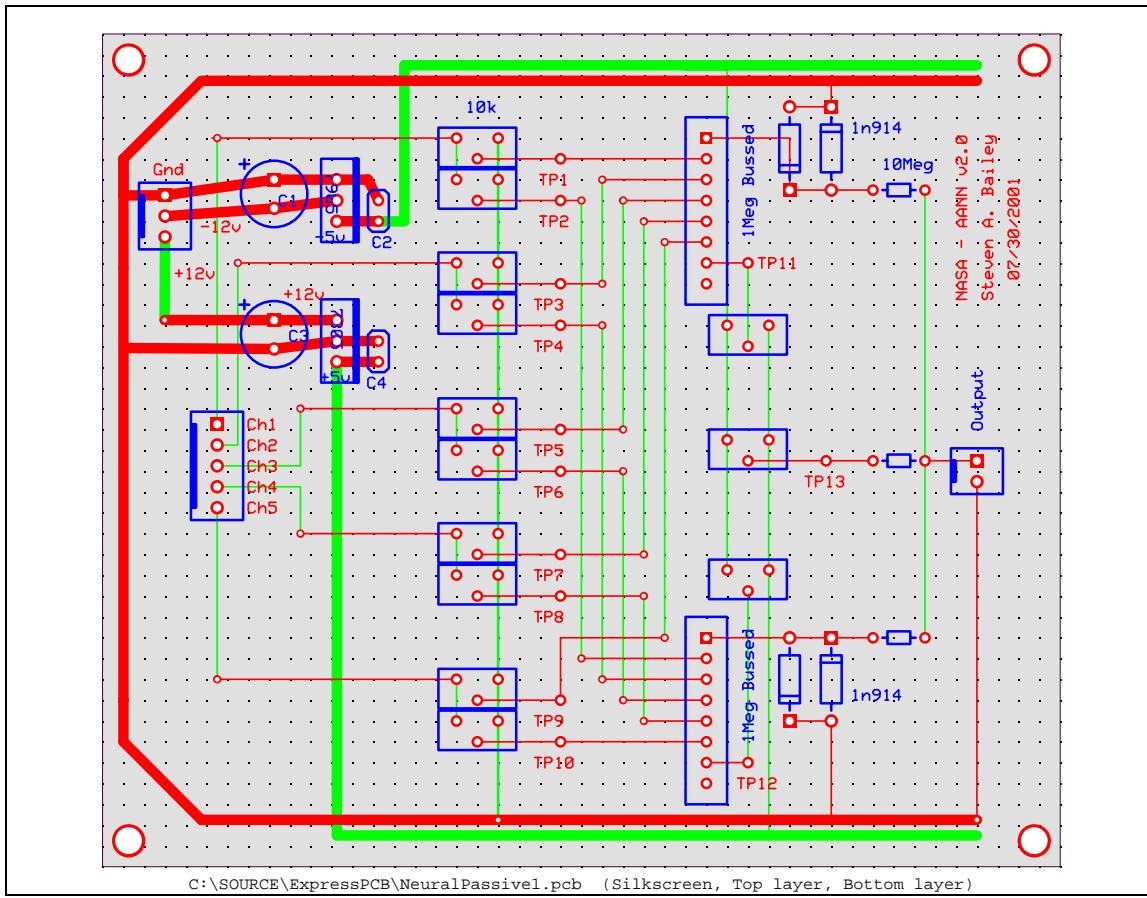


Figure 29 - Passive AANN PCB

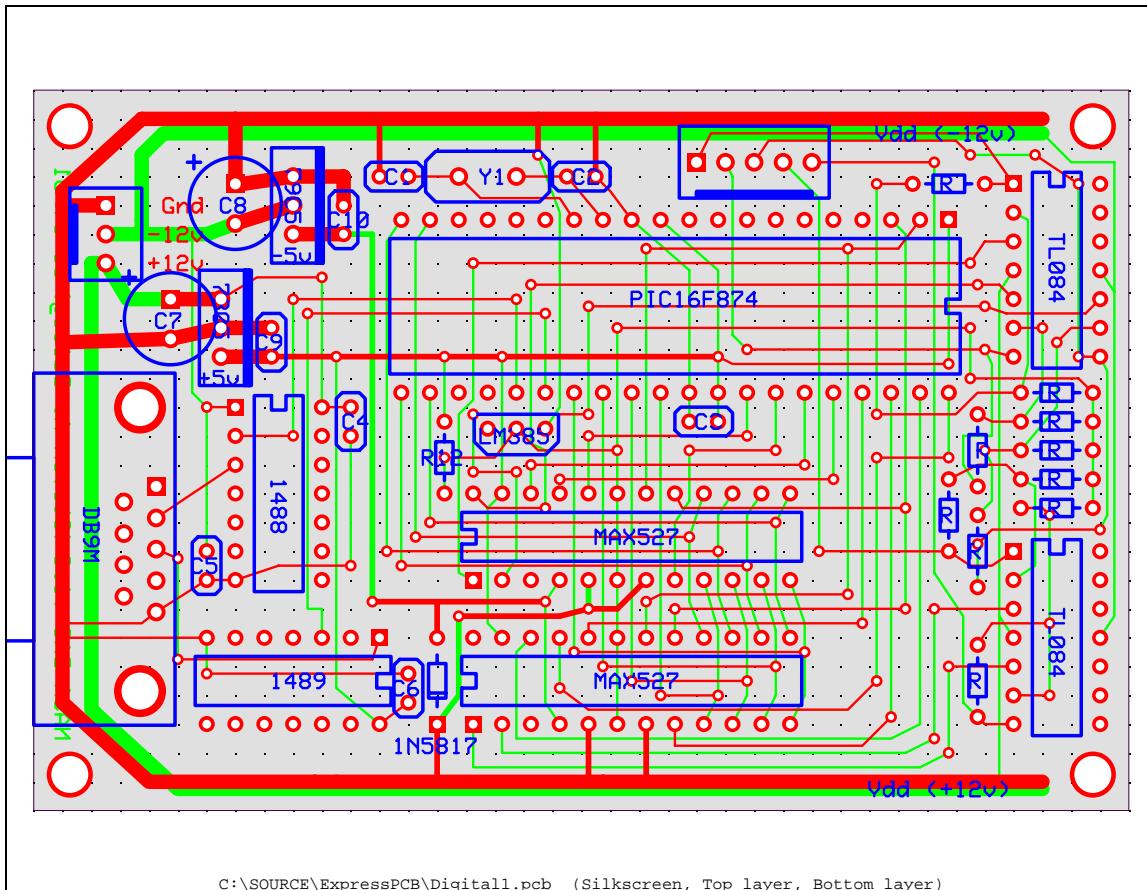


Figure 30 – Digital-to-Analog (DAC) PCB